

· 大数据时代的R语言 ·

数据分析： R语言实战

李诗羽 张飞 王正林 编著

电子工业出版社

Publishing House of Electronics Industry

北京•BEIJING

内 容 简 介

大数据时代,数据成为决策最为重要的参考之一,数据分析行业迈入了一个全新的阶段。**R**是一款非常优秀的统计分析软件,本书侧重于使用**R**进行数据的处理、整理和分析,重点讲述了**R**的数据分析流程、算法包的使用以及相关工具的应用,同时结合大量精选的数据分析问题对**R**软件进行科学、准确和全面的介绍,以便使读者能深刻理解**R**的精髓和灵活、高效的使用技巧。

通过本书,读者不仅能掌握使用**R**及相关的算法包来快速解决实际问题,而且能学会从实际问题分析入手,到利用**R**进行求解,以及对结果进行分析。

本书可作为计算机、互联网、机器学习、信息、数学、经济金融、管理、运筹、统计以及相关理工科专业的本科生、研究生的学习用书,也能帮助市场营销、金融、财务、人力资源管理人员及产品经理解决实际问题,还能帮助从事咨询、研究、分析行业的人士及各级管理人员提高专业水平。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

数据分析: R语言实战 / 李诗羽, 张飞, 王正林编著. —北京: 电子工业出版社, 2014.8

(大数据时代的 R 语言)

ISBN 978-7-121-23714-0

I. ①数… II. ①李… ②张… ③王… III. ①统计数据—统计分析②程序语言—程序设计 IV. ①O212.1
②TP312

中国版本图书馆 CIP 数据核字(2014)第 147847 号

策划编辑: 张月萍

责任编辑: 刘 舫

印 刷: 三河市鑫金马印装有限公司

装 订: 三河市鑫金马印装有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编: 100036

开 本: 787×980 1/16 印张: 21.00 字数: 521 千字

版 次: 2014 年 8 月第 1 版

印 次: 2014 年 8 月第 1 次印刷

定 价: 59.00 元

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010)88254888。

质量投诉请发邮件至 zltts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010)88258888。

前言

大数据时代，数据成为决策最为重要的参考之一，数据分析随着大数据概念的普及而日益得到重视，数据分析行业迈入了一个全新的阶段。

数据分析的软件如雨后春笋般地涌现，其中 R 软件的发展备受瞩目。R 是一个免费开源软件，它提供了首屈一指的统计计算和绘图功能，尤其是大量的统计分析、数据挖掘方面的算法包，使得它成为一款优秀的、不可多得的数据分析工具软件。

本书的主要目的是向读者介绍如何用 R 进行数据分析，通过大量的精选实例，循序渐进、全面系统地讲述 R 在数据分析领域的应用。

全书分为 15 章，主要内容从数据分析的前期准备、基本分析及应用和综合实例这三篇展开。

（1）上篇 数据分析的前期准备

由第 1~3 章组成，首先简要介绍数据分析的原则、步骤和过程，常用工具及 R 在数据分析中的优势，然后介绍 R 中数据整理等数据预处理的基本函数及方法。这些内容是使用 R 进行数据分析的最基础内容。

（2）中篇 基本分析及应用

由第 4~13 章组成，主要讲述数据分析的基本算法及应用，包括数据的图形描述、描述性分析、参数估计、假设检验，以及方差分析、回归分析、主成分分析、典型相关分析、判别分析、聚类分析和时间序列分析等，这些分析方法也是数据分析中使用得最多、最普遍的算法。R 中提供了丰富的、功能强大的算法包和实现函数，数据分析的初级和中级用户务必掌握。

（3）下篇 综合实例

由第 14~15 章组成，主要结合两个大例子，综合讲述数据分析在金融数据分析和数据预测中的应用，以及如何使用 R 中的方法和工具进行应用。对于中高级的用户，可以深入学习一下。

R 的特点是入门非常容易，使用也非常简单，因此本书也不需要读者具备 R 和数据挖掘的基础知识，不管是 R 初学者，还是熟练的 R 用户都能从书中找到对自己有用的内容，从而快速入门和提高。读者既可以把本书作为学习如何应用 R 的一本优秀教材，也可以作为数据分析的工具书。

全书以实际问题、解决方案和对解决方案的讨论为主线来组织内容，脉络清晰，并且各章自

成体系。读者可以从头至尾逐章学习，也可以根据自己的需要进行学习，找到自己实际问题的解决方案。

本书所编的源程序，都通过了反复的调试，读者可在 www.broadview.com.cn/23714 网站下载，方便读者使用。

本书主要由李诗羽、张飞、王正林编写，其他参与编写的人员有肖静、邹术来、夏路生、钟救元、郑曙霞、王成、刘亚文、肖绍英、王伟欣、朱桂莲、夏立德、王龙跃等。在此对所有参与编写的人员表示感谢！对关心、支持我们的读者表示感谢！

由于时间仓促，作者水平和经验有限，书中错漏之处在所难免，敬请读者指正，我们的电子邮箱是：wa_2003@126.com。

编著者

2014 年 5 月 28 日于北京

目 录

第 0 章 致敬，R！	1
致敬，肩膀！	1
致敬，时代！	3
致敬，人才！	3
致敬，R 瑟！	5

上篇 数据分析的前期准备

第 1 章 数据分析导引	8
1.1 数据分析概述	8
1.1.1 数据分析的原则	8
1.1.2 数据分析的步骤	9
1.1.3 数据分析的过程	10
1.1.4 数据分析的对象	11
1.2 大数据分析	11
1.2.1 大数据分析的流程	11
1.2.2 大数据分析的基本方面	12
1.2.3 大数据分析的应用	13
1.3 数据分析常用工具	13
1.4 R 在数据分析中的优势	14
第 2 章 数据的读取与保存	16
2.1 数据读取	16
2.1.1 读取内置数据集	16
2.1.2 读取文本文件	17
2.1.3 读取固定宽度格式的文件	20
2.1.4 读取 Excel 数据	21
2.1.5 读取数据库文件	22
2.1.6 读取网页数据	26
2.1.7 读入 R 格式的文件	28
2.1.8 从其他统计软件读入数据	28
2.2 数据保存	31
2.2.1 使用函数 cat()	31

2.2.2	保存为文本文件	32
2.2.3	保存 R 格式文件	33
2.2.4	保存为其他类型文件	33
第 3 章	数据预处理	34
3.1	基本函数	34
3.2	数据修改	38
3.2.1	修改数据标签	38
3.2.2	行列删除	38
3.3	缺失值处理	38
3.3.1	判断缺失数据	39
3.3.2	判断缺失模式	39
3.3.3	处理缺失数据	41
3.4	数据整理	44
3.4.1	数据合并	44
3.4.2	选取数据的子集	46
3.4.3	数据排序	47
3.5	长宽格式的转换	48
3.5.1	揉数据函数	48
3.5.2	揉数据的最佳伴侣	49

中篇 基本分析及应用

第 4 章	数据的图形描述	54
4.1	R 绘图概述	54
4.2	绘图区域分割	55
4.2.1	函数 par()	55
4.2.2	函数 layout()	56
4.2.3	函数 split.screen()	57
4.3	二维图形	58
4.3.1	高级绘图函数	58
4.3.2	多元数据绘图	61
4.3.3	低级绘图函数	63
4.3.4	图形美化	64
4.3.5	交互式绘图命令	65
4.4	三维图形	67
4.5	lattice 程序包	69
4.6	ggplot2 程序包	73
4.6.1	快速绘图	74
4.6.2	分图层绘图	76

4.7 图形保存.....	84
4.8 实战实例：数据地图.....	84
第 5 章 数据的描述性分析	88
5.1 R 内置的分布.....	88
5.2 集中趋势的分析.....	90
5.2.1 集中趋势的测度.....	90
5.2.2 R 语言实现.....	91
5.3 离散趋势的分析.....	93
5.3.1 离散趋势的测度.....	93
5.3.2 R 语言实现.....	94
5.4 数据的分布分析.....	95
5.4.1 分布情况的测度.....	95
5.4.2 R 语言实现.....	96
5.5 图形分析及 R 实现.....	97
5.5.1 直方图和密度函数图.....	97
5.5.2 QQ 图.....	98
5.5.3 茎叶图.....	100
5.5.4 箱线图.....	100
5.5.5 经验分布图.....	102
5.6 多组数据分析及 R 实现.....	102
5.6.1 多组数据的统计分析.....	102
5.6.2 多组数据的图形分析.....	103
第 6 章 参数估计及 R 实现	112
6.1 点估计及 R 实现.....	112
6.1.1 矩估计.....	112
6.1.2 极大似然估计.....	116
6.2 单正态总体的区间估计.....	122
6.2.1 均值 μ 的区间估计.....	122
6.2.2 方差 σ^2 的区间估计.....	125
6.3 两正态总体的区间估计.....	126
6.3.1 均值差 $\mu_1 - \mu_2$ 的区间估计.....	127
6.3.2 两方差比 σ_1^2 / σ_2^2 的区间估计.....	130
6.4 关于比率的区间估计.....	131
第 7 章 假设检验及 R 实现	134
7.1 假设检验概述.....	134
7.1.1 理论依据.....	135
7.1.2 检验步骤.....	135

7.1.3 两类错误	136
7.2 单正态总体的检验	137
7.2.1 均值 μ 的检验	138
7.2.2 方差 σ^2 的检验	141
7.3 两正态总体的检验	142
7.3.1 均值差 $\mu_1 - \mu_2$ 的检验	143
7.3.2 成对数据的 t 检验	146
7.3.3 两总体方差的检验	147
7.4 比率的检验	148
7.4.1 比率的二项分布检验	148
7.4.2 比率的近似检验	149
7.5 非参数的检验	149
7.5.1 总体分布的 χ^2 检验	150
7.5.2 Kolmogrov-Smirnov 检验	153
第 8 章 方差分析及 R 实现	157
8.1 单因素方差分析及 R 实现	157
8.1.1 基本假设的检验	157
8.1.2 单因素方差分析	160
8.1.3 多重 t 检验	164
8.1.4 Kruskal-Wallis 秩和检验	166
8.2 双因素方差分析及 R 实现	168
8.2.1 无交互作用的分析	169
8.2.2 有交互作用的分析	172
8.3 协方差分析及 R 实现	176
第 9 章 回归分析及 R 实现	180
9.1 一元线性回归	180
9.1.1 模型理论	180
9.1.2 显著性检验	181
9.1.3 R 语言实现	181
9.2 多元线性回归	187
9.2.1 模型理论	187
9.2.2 显著性检验	188
9.2.3 R 语言实现	189
9.2.4 逐步回归	192
9.3 回归诊断及 R 实现	194
9.3.1 残差诊断	195
9.3.2 影响分析	198
9.3.3 多重共线性诊断	201

9.4	岭回归及 R 实现	203
9.5	广义线性模型	206
9.5.1	模型理论	206
9.5.2	R 语言实现	207
第 10 章	主成分分析与因子分析	211
10.1	主成分分析	211
10.1.1	理论基础	211
10.1.2	R 语言实现	215
10.2	因子分析	221
10.2.1	理论模型	221
10.2.2	因子载荷矩阵的估计方法	223
10.2.3	R 语言实现	225
第 11 章	典型相关分析和对应分析	230
11.1	典型相关分析	230
11.1.1	理论基础	230
11.1.2	典型相关分析的应用	232
11.1.3	R 语言实现	233
11.2	对应分析	236
11.2.1	理论基础	236
11.2.2	对应分析的步骤	237
11.2.3	R 语言实现	238
第 12 章	判别分析和聚类分析	242
12.1	判别分析及 R 实现	242
12.1.1	距离判别法	243
12.1.2	距离判别法的 R 实现	244
12.1.3	Fisher 判别法	247
12.1.4	Fisher 判别法的 R 实现	248
12.1.5	贝叶斯判别法	251
12.1.6	贝叶斯判别法的 R 实现	252
12.2	聚类分析及 R 实现	252
12.2.1	理论概述	253
12.2.2	R 实现举例	254
第 13 章	时间序列分析及 R 实现	260
13.1	时间序列的基本分析	260
13.1.1	平稳性与非平稳性	260
13.1.2	R 实现的基本步骤	261
13.2	时间序列的分解	262

13.2.1	分解非季节性数据	263
13.2.2	分解季节性数据	265
13.3	指数平滑法预测分析	268
13.3.1	简单指数平滑法	269
13.3.2	残差的白噪声检验	272
13.3.3	Holt 指数平滑法	275
13.3.4	Winters 指数平滑法	277
13.4	ARIMA 模型分析	280
13.4.1	基本思想	280
13.4.2	平稳化处理	281
13.4.3	建模	282
13.4.4	模型的参数估计	284
13.4.5	模型预测及检验	284

下篇 综合实例

第 14 章	R 在金融数据分析中的应用	288
14.1	投资组合最优化实例	288
14.1.1	概述	288
14.1.2	均值-方差模型	289
14.1.3	模拟退火算法	292
14.2	构造投资组合的有效前沿	298
14.2.1	R 中的算法包	298
14.2.2	计算分析	298
14.3	股票聚类分析	301
14.3.1	概述	301
14.3.2	K-means 聚类分析	302
14.3.3	层次聚类分析	304
第 15 章	R 在数据预测中的应用	306
15.1	回归分析预测	306
15.1.1	概述	306
15.1.2	实战案例	306
15.2	时间序列预测	318
15.2.1	概述	318
15.2.2	实战案例	318

第 0 章

致敬，R！

此时，你一定想知道，书的封面上停着一只什么鸟？

那我告诉你，那是 Robin 鸟，中文名叫知更鸟，它可大有来头，是英国的国鸟，以羽毛颜色漂亮招人喜爱著称。

我把它放在封面，首先是借用其名字首字母 R，来表示 R 语言。最重要的是，我想到了股神巴菲特的一句关于知更鸟的名言，我想双关暗示一下——如果你还不学一些 R，大数据对你来说就快结束了。

如果你想等到知更鸟报春，那春天就快结束了。——巴菲特

So if you wait for the robins, spring will be over. ——Warren Edward Buffett

如果你想快速成功

你最好站在一个高的肩膀上

如果你想驾驭大数据时代

你最好懂点数据挖掘

如果你想玩转数据挖掘

你最好先玩转 R！

致敬，肩膀！

可能当我们还是三好小学生的时候，我们就知道，牛顿是站在巨人的肩膀上的，现如今，我们都知道，中国所有的“二代”，不是站在老爹的肩膀上，就是踩在老丈人的肩膀上的。不得不承认，脚下的肩膀有时候是很牛的。

当你走进数据挖掘，当你走进 R 的世界，你会发现，R 的脚下也有一个肩膀，有肩膀的 R 也

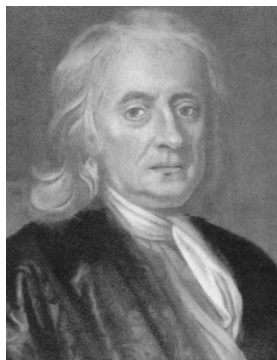
是很牛的！

R 的肩膀，是谷歌首席经济学家范里安先生发现的，先生说了好几句话，我只记住了这句“使用 R，你已经站在了巨人的肩膀上”。

在此，我只想致敬一下肩膀，与“二代”无关！

我之所以能取得现在的成就，是因为我站在巨人的肩膀上。——牛顿

If I have seen further it is by standing on the shoulders of giants. ——Isaac Newton



艾萨克·牛顿爵士 (Isaac Newton, 1643.12.25—1727.3.20)，
英国数学家、物理学家、天文学家和经典力学体系奠基人。

R 的最美之处在于，你能够通过修改很多牛人预先编写好的包的代码，解决你想解决的各种问题，因此，事实上，使用 R，你已经站在了巨人的肩膀上。——哈尔·罗纳德·范里安

The great beauty of R is that you can modify it to do all sorts of things. And you have a lot of prepackaged stuff that's already available, so you're standing on the shoulders of giants.

——Hal Ronald Varian



哈尔·罗纳德·范里安 (Hal Ronald Varian)，
谷歌首席经济学家，美国著名研究微观经济学和信息经济学学者。

致敬，时代！

“大数据”一词，最早是全球知名咨询公司麦肯锡提出来的，“数据，已经渗透到当今每一个行业和业务职能领域，成为重要的生产因素。人们对于海量数据的挖掘和运用，预示着新一波生产率增长和消费者盈余浪潮的到来。”

我们，已经身处大数据时代了，对于做数据挖掘、用 R 的我们来说，好时代来了！

“大数据”时代已经降临，在商业、经济及其他领域中，决策将日益基于数据和分析而做出，而并非基于经验和直觉。

——摘自《纽约时报》，2012 年 2 月的一篇专栏



摘自《纽约时报》，How Big Data Became So Big 一文

“在美国具备高度分析技能的人才（大学及研究生院中学习统计和机器学习专业的学生）供给量，2008 年为 15 万人，预计到 2018 年将翻一番，达到 30 万人。然而，预计届时对这类人才的需求将超过供给，达到 44 万~49 万人的规模，这意味着将产生 14 万~19 万的人才缺口。仅仅四五年前，对数据科学家的需求还仅限于 Google、Amazon 等互联网企业中。然而在最近，重视数据分析的企业，无论是哪个行业，都在积极招募数据科学家，这也令人手不足的状况雪上加霜。”

——摘自麦肯锡全球研究院的报告 Big data: The next frontier for innovation, competition and productivity（大数据：未来创新、竞争、生产力的指向标），2011.5

……2017 年大数据技术和服务市场将增至 324 亿美元，实现 27% 的年复合增长率。……大数据不仅是新兴行业，也是市场的主要驱动力，它正在酿成一个主要的市场。

——摘自国际数据公司 IDC 的预测报告 Worldwide Big Data Technology and Services 2013–2017 Forecast, 2013.12

致敬，人才！

Google 首席经济学家范里安先生，在 2008 年 10 月与麦肯锡总监 James Manyika 先生的对话

中，曾经讲过下面一段话：“我总是说，在未来 10 年里，从事最有意思的工作的人将是统计学家。人们都认为我在开玩笑。但是，过去谁能想到电脑工程师会成为 20 世纪 90 年代从事最有趣的工作的人？在未来 10 年里，获取数据——以便能理解它、处理它、从中提取价值、使其形象化、传送它——的能力将成为一种极其重要的技能，不仅在专业层面上是这样，而且在教育层面（包括对中小学生、高中生和大学生的教育）也是如此。由于如今我们已真正拥有实质上免费的和无所不在的数据，因此，与此互补的稀缺要素是理解这些数据并从中提取价值的能力。”

范里安教授在当初的对话中使用的是 *statisticians*（统计学家）一词，虽然当时他没有使用数据科学家这个词，但这里所指的，正是现在我们普遍所指的数据科学家。

对数据科学家的关注，源于大家逐步认识到，Google、Amazon、Facebook 等公司成功的背后，存在着这样一批专业人才。这些互联网公司对于大量数据不是仅进行存储而已，而是将其变为有价值的金矿——例如，搜索结果、定向广告、准确的商品推荐、可能认识的好友列表等。

仅仅在几年前，数据科学家还不是一个正式确定的职业，然而一眨眼的工夫，这个职业就已经被誉为“今后 10 年 IT 行业最重要的人才”了。



摘自 The Emerging Role of the Analyst 一文

在国外，据统计，目前世界 500 强企业中，有 90% 以上都建立了数据分析部门。IBM、微软、Intel 等公司也积极投资数据业务，建立大数据部门，培养数据分析团队。

美国的小伙伴们，在数据挖掘、数据科学等方面比我们下手早。2011 年，美国的加州大学伯克利分校开始开设《数据科学导论》课程；伊利诺伊大学香槟分校从 2011 年起举办“数据科学暑期研究班”；哥伦比亚大学从 2013 年起开设《应用数据科学》课程，并从 2013 年起开设相关培训项目，还计划从 2014 年起设立硕士学位，2015 年设立博士学位；纽约大学从 2013 年秋季起设立“数据科学”硕士学位；在英国，邓迪大学从 2013 年起设立“数据科学”硕士学位……

怎么办，那就自学吧，从 R 开始，站上那个肩膀，做今后 10 年最重要的人才吧！

致敬, R 瑟!

1976 年, John Chambers 在贝尔实验室开发的 S 语言是为了替代昂贵的 SPSS 和 SAS 工具。如果说 S 是 VAX 和 UNIX 小型机时代的产物, 那么 R 则是 PC 和 Linux 时代的产物, R 语言大量借用了 S 语言的方法。

1992 年, 新西兰奥克兰大学的两位统计学教授, 两位“R 姓”先生(R Sir, “R 瑟”) Ross Ihaka 和 Robert Gentleman 成为了同事, 为了方便教授初等统计课程, 这哥儿俩开发了一种语言, 而恰巧他们名字的首字母都是 R, 于是 R 便成为这门语言的名称。

这两位 R 教授也是 R 开发团队的核心成员, 值得注意的是, S 语言的发明者 John Chambers 也是 R 开发团队的成员, 因此不难理解 R 语言的一些数据处理路径与 S 语言相同。

R 可以看作 S 的一种实现, Insightful 公司开发的 S-PLUS 也是 S 的实现版本, 2004 年 Insightful 把 S-PLUS 授权给了朗讯科技, 后来又被 Tibco 软件于 2008 年收购。



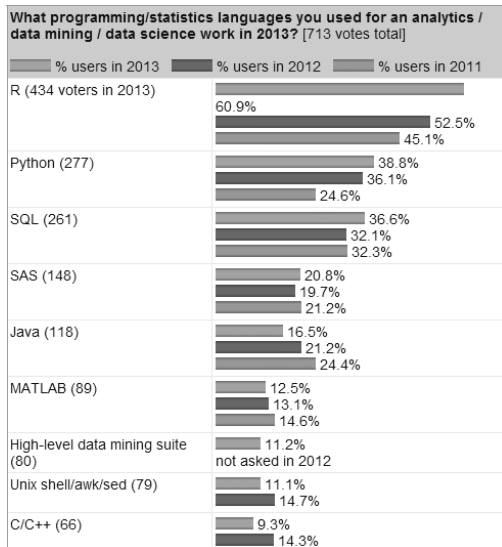
R 语言的发明者 Ross Ihaka 和 Robert Gentleman

与 S 和 S-PLUS 不同的是, R 并不是象牙塔里炮制出的代码, 而是一个由分析师和程序员构成的社区的产物, 这个社区为处理各种数据集创建了超过 5000 个函数包和 2500 个插件。

今天, 根据 Revolution Analytics 的统计, R 被全球超过 200 万个量化分析师采用。Revolution Analytics 成立于 2007 年, 并开发出了 R 的并行实现, 该公司采用了开放内核的方式开发 R, 为开源软件包推广商业支持, 同时扩展 R 环境, 提升其在计算机集群上的表现, 并将其与 Hadoop 集群对接。

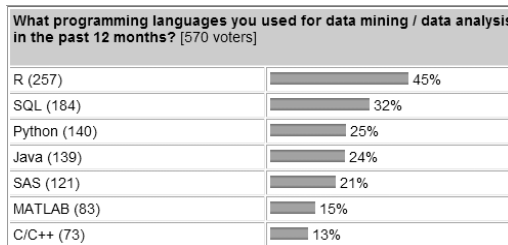
在 2013 年中, 数据挖掘专业网站 KDnuggets 做了一个关于“什么样的程序或者统计语言是你在做分析、挖掘、科学计算的时候所需要的?”的调查。

调查结果是: 最受欢迎的是 R 语言(61%的调研会员在用), 然后是 Python(39%)、SQL(37%)等, 每个调研对象平均使用 2~3 种语言。

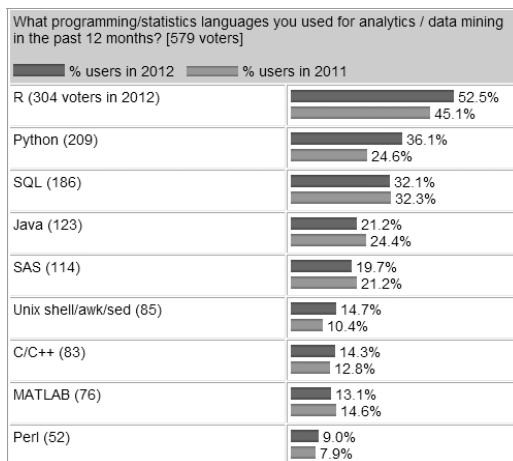


2013 年 KDnuggets 的调查结果

R 位列最受欢迎的数据挖掘软件，其实不足为奇，因为它已经三连冠了！



2012 年 KDnuggets 的调查结果



2011 年 KDnuggets 的调查结果

上篇

数据分析的前期准备

- 第 1 章 数据分析导引
- 第 2 章 数据的读取与保存
- 第 3 章 数据预处理

第 1 章

数据分析导引

数据分析这一学科已成为统计学、机器学习等诸多领域的研究热点，数据分析技术也已成为大数据时代热门的技术。

数据分析近年来发展异常迅猛，由此不仅产生了大量不同类型、功能强大的统计分析算法，而且也推动了众多数据分析工具软件的发展，在这些软件当中，R 已悄然成为了数据分析领域最重要的软件之一。

R 是一个包含众多科学、工程统计的庞大系统，是目前世界上最流行的统计软件之一。R 既是用于统计计算和统计制图的优秀工具，又是大数据分析和挖掘重要的工具。

1.1 数据分析概述

数据分析是指用适当的统计方法对收集来的大量第一手资料和二手资料进行分析，以求最大化地开发数据资料的功能，发挥数据的作用。数据分析是为了提取有用信息和形成结论而对数据加以详细研究和概括总结的过程。

1.1.1 数据分析的原则

数据分析必须遵循的原则主要有三条：

(1) 数据分析是为了验证假设的问题，需要提供必要的验证数据。在数据分析中，分析模型构建完成后，需要利用测试数据验证模型的正确性。

(2) 数据分析是为了挖掘更多的问题，并找到深层次的原因。比如你分析产品销售情况的数据，你要找到为什么销售数据会变动，由浅入深地分析其原因，比如说是促销、节日、天气、卖场宣传、卖场环境、消费心理、消费人群、价格、竞争对手等原因。然后，继续总结，深入分析，

针对可能的原因进行实际运用及跟踪结果，之后再分析。

(3) 不能为了做数据分析而做数据分析。如果没有明确的问题或者目标，你拿着手中的数据，就一头扎进去进行数据分析，这是不会有结果的。进行一次数据分析的原因，一定是你发现了数据反映了一些问题，或者是你希望通过分析，达到改善某方面的问题，比如说产品的销量、顾客的流量等。

因此，进行数据分析之前，要尽量明确数据分析针对的问题，然后带着问题进行数据分析。因为面对不同的问题，分析思路、分析方法甚至结果都可能大相径庭。

你可以先围绕数据，大概地问一堆问题，然后从中确定你要分析的问题，比如你希望通过数据分析来改善什么？目前通过网络营销带来的销售量有什么问题？流量最多能提升多少？网站主要针对的人群是什么？现在的转化率是否正常？等等。这些看似很不重要的问题，其实你可以从客户的回答中得到很多答案，从而来确定你要分析的问题。

1.1.2 数据分析的步骤

数据分析有极广泛的应用范围。典型的数据分析包含以下三个基本步骤。

(1) 探索性数据分析

当你从多种渠道获得了大量的可能杂乱无章、看不出规律的数据的时候，首先需要在没有多少经验的情况下第一次对其进行仔细的分析。探索性数据分析（EDA: Exploratory Data Analysis）能够在这种情况下帮助你找到所有这些数据中隐含的信息。

不仅如此，探索性数据分析还能够引导你建立有用的模型，比如通过作图、造表、用各种形式的方程拟合、计算某些特征量等手段探索规律性的可能形式，即往什么方向和用何种方式去寻找和揭示隐含在数据中的规律性。

即使在你类似的数据已经司空见惯，或者认为建模工作只是例行公事的情况下，事先运用探索性数据分析方法检验模型的可靠性、验证相关的假设也是一项非常有用和重要的工作，它往往能使你获得意想不到的发现。

本质而言，探索性数据分析是启发式、开放式和完全动态的。使用这种方法的时候常常也需要对数据进行清洗和整合，这些工作非常有助于你运用多种可视化的方法真正实现“让数据说话”。

(2) 模型选定分析

在探索性分析的基础上，通过定量分析方法，提出一类或几类可能的模型，然后通过进一步的分析，从中挑选一类适合的模型。

(3) 推断分析

通常使用数理统计方法，进行一系列的计算和分析，对所确定的模型或估计的可靠程度和精

确程度做出推断。

1.1.3 数据分析的过程

数据分析的过程通常包括明确目标、收集数据、加工整理、选择方法和解释结果。

（1）明确目标

明确目标是数据分析的出发点。明确数据分析的目标就是要明确本次数据分析要研究的主要问题和预期的分析目标等。例如，分析储户的储蓄行为是否存在显著差异以及原因；分析某电子商务网站的客户群特征，包括其人口特征和消费行为等方面。只有明确了数据分析的目标，才能正确地制定数据收集方案，即收集哪些数据，采用怎样的方式收集等，进而为数据分析做好准备。

（2）收集数据

收集数据当然是要正确地收集数据，正确的数据对于实现数据分析目标将起到关键性的作用。正确收集数据是指从分析目标出发，排除干扰因素，正确收集服务于既定分析目标的数据。

排除数据中那些与目标不关联的干扰因素是数据收集中的重要环节。数据分析并不仅仅是对数据进行数学建模，收集的数据是否真正符合数据分析的目标，其中是否包含了其他因素的影响，影响程度怎样，应如何剔除这些影响等问题都是数据分析过程中必须注意的重要问题。

（3）加工整理

在明确数据分析目标基础上收集到的数据，往往还需要对其进行必要的加工整理，而后才能真正用于分析建模。

数据的加工整理通常包括数据缺失值处理、数据的分组、基本描述统计量的计算、基本统计图形的绘制、数据取值的转换、数据的正态化处理等，它能够帮助人们掌握数据的分布特征，这是进一步深入分析和建模的基础。

（4）选择方法

数据加工整理完成后一般就可以进行进一步的数据分析了。分析时应切忌滥用和误用统计分析方法。滥用和误用统计分析方法主要是由于对方法能解决哪类问题、方法适用的前提、方法对数据的要求不清等原因造成的。另外，统计分析软件的不断普及和应用中的不求甚解也会加重这种现象。因此，在数据分析中应避免盲目的“拿来主义”，否则，得到的分析结论可能会偏差较大甚至发生错误。

另外，选择几种统计分析方法对数据进行探索性的反复分析也是极为重要的。每一种统计分析方法都有自己的特点和局限性，因此，一般需要选择几种方法反复印证分析，仅依据一种分析方法的结果就断然下结论是不科学的。

（5）解释结果

数据分析的直接结果是统计量和统计参数。正确理解它们的统计含义是得出一切分析结论的基础，它不仅能帮助人们有效避免毫无根据地随意引用统计数字的错误，同时也是证实分析结论正确性和可信性的依据，而这一切都取决于人们能否正确地把握统计分析方法的核心思想。

另外，将统计量和统计参数与实际问题相结合也是非常重要的。数据分析方法是否能够正确地解决各学科的具体问题，不仅取决于应用分析方法或工具的人能否正确地选择分析方法，还取决于他们是否具有深厚的应用背景。只有将各学科的专业知识与统计量和统计参数相结合，才能得出令人满意的分析结论。

1.1.4 数据分析的对象

常见的数据分析的对象有以下 7 大类：

- 关系型数据库、事务型数据库、面向对象的数据数据库
- 数据仓库 / 多维数据库
- 空间数据（如地图信息）
- 工程数据（如建筑、集成电路的信息）
- 文本和多媒体数据（如文本、图像、音频、视频数据）
- 时间相关的数据（如历史数据或股票交易数据）
- 万维网（如半结构化的 HTML、结构化的 XML 以及其他网络信息）

1.2 大数据分析

1.2.1 大数据分析的流程

相比于传统的数据处理，大数据时代的数据处理的理念有三大显著的转变：首先，数据是全体的而不是抽样的；其次，分析要的是效率而不是绝对精确；第三，分析的结果要的是相关性而不是因果性。

常见的大数据处理流程，可以概括为四步：数据采集、预处理、统计和分析以及数据挖掘。

（1）数据采集

大数据的采集主要是指利用多个数据库来接收发自客户端的数据，并且用户可以通过这些数据库来进行简单的查询和处理工作。

（2）预处理

虽然采集端本身会有很多数据库，但是如果要对这些海量数据进行有效的分析，还是应该将这些来自前端的数据导入到一个集中的大型分布式数据库，或者分布式存储集群中，并且可以在

导入的基础上做一些简单的清洗和预处理工作。

（3）统计和分析

统计和分析主要利用分布式数据库或者分布式计算集群来对存储于其内的海量数据进行普通的分析和分类汇总等，以满足大多数常见的分析需求。

统计和分析这个环节的主要特点和挑战是分析涉及的数据量大，其对系统资源，特别是 I/O 会有极大的占用。

（4）数据挖掘

与前面统计和分析过程不同的是，数据挖掘一般没有什么预先设定好的主题，主要是在现有数据上面进行基于各种算法的计算，起到预测的效果，从而实现一些高级别数据分析的需求。数据挖掘的特点和挑战主要是由于挖掘的算法很复杂，并且计算涉及的数据量和计算量都很大。

1.2.2 大数据分析的基本方面

越来越多的应用涉及大数据，这些大数据的属性，包括数量、速度和多样性等都呈现了大数据不断增长的复杂性，因此，大数据的分析方法在大数据领域就显得尤为重要，可以说是决定最终信息是否有价值的决定性因素。大数据分析的基本方面可概括如下。

（1）预测性分析能力

预测性分析可以让分析员根据可视化分析和数据挖掘的结果做出一些预测性的判断，在此基础上，进一步的数据分析、数据挖掘可以让分析员更好地理解数据。

（2）数据质量和数据管理

数据质量和数据管理是一些管理方面的最佳实践。通过标准化的流程和工具对数据进行处理，可以保证一个预先定义好的高质量的分析结果。

（3）可视化分析

不管是对数据分析专家还是普通用户，数据可视化是数据分析工具最基本的要求。可视化可以直观地展示数据，让数据自己说话，让观众看到结果。

（4）语义引擎

大数据中非结构化的数据日益增多，非结构化数据的多样性带来了数据分析新的挑战，需要一系列的工具去解析、提取及分析数据。语义引擎需要被设计成能够从“文档”中智能提取信息。

（5）数据分析挖掘算法

可视化是给人看的，数据分析挖掘就是给机器看的。集群分析、分割分析、孤立点分析还有其他的算法让我们可以深入数据内部，挖掘价值。这些算法不仅要处理大数据的量，而且也要有

处理大数据的速度。

1.2.3 大数据分析的应用

当前，大数据分析应用的主要领域有以下这些。

（1）理解和定位客户

这是目前最大、最广为人知的大数据应用领域之一。这里的重点是使用大数据来更好地了解客户以及他们的行为和喜好。企业都热衷于收集社交媒体数据、浏览器日志、文本分析和传感器数据，来更全面地了解他们的客户。在大多数情况下，总的目标是创建预测模型。

例如美国零售商 Target 利用大数据分析，可以非常准确地预测他们的客户什么时候想要小孩。另外，通过使用大数据，电信公司可以更好地预测客户流失，沃尔玛可以更好地预测哪些产品将会热卖，汽车保险公司能够了解其客户的驾驶水平。

（2）理解和优化业务流程

大数据也越来越多地用于优化业务流程，比如通过利用从社交媒体数据、网络搜索趋势以及天气预报挖掘出的预测信息，零售商能够优化其库存。其中广泛应用大数据分析的业务流程是供应链或配送路线优化。在这方面，地理定位或无线电频率识别传感器被用来追踪货物或送货车，并通过整合实时交通数据来优化路线。

（3）金融交易

大数据在金融行业的应用主要是金融交易。高频交易（HFT）是大数据应用比较多的领域。其中，大数据算法被用来做出交易决定。现在，大多数股权交易都是通过大数据算法进行的，这些算法越来越多地开始考虑社交媒体网络和新闻网站的信息，以便在几秒内做出买入和卖出的决定。

1.3 数据分析常用工具

数据分析工作在绝大多数情况下的目的在于用统计学的手段揭示数据反映的一些有用的信息，比如事物的发展趋势和规律；又或者是定位某种或某些现象的原因；也可以是检验某种假设是否正确（心智模型的验证）。

用于数据分析的工具很多，常用的工具有 Excel、SPSS、Matlab 和 R 等。

（1）Excel

Excel 软件大多数人应该都是比较熟悉的，它满足了绝大多数办公制表的需求，同时也拥有相当优秀的数据处理能力。其自带的 ToolPak（分析工具库）和 Solver（规划求解加载项）可以完成基本描述统计、方差分析、统计检验、傅里叶分析、线性回归分析和线性规划求解工作。这些功能在 Excel 中没有默认打开，需要在 Excel 选项中手动开启。

除此以外，Excel 也提供较为常用的统计图形绘制功能。这些功能涵盖了基本的统计分析手段，其已经能够满足绝大部分数据分析工作的需求，同时也提供了相当友好的操作界面，对于具备基本统计学理论的用户来说是十分容易上手的。

（2）SPSS

SPSS 原名是 Statistical Package for the Social Sciences（社会科学统计软件包），现在已被 IBM 收购，改名后仍然叫 SPSS，不过全称变更为 Statistical Product and Service Solutions（统计产品与服务解决方案）。

SPSS 是一个专业的统计分析软件，除了基本的统计分析功能之外，还提供非线性回归、聚类分析（Clustering）、主成分分析（PCA）和基本的时序分析。SPSS 在某种程度上可以进行简单的数据挖掘工作，比如 K-Means 聚类，不过数据挖掘的主要工作一般都是使用其自家的 Clementine（现已改名为 SPSS Modeler）完成，而且 SPSS Modeler 的建模功能非常强大且智能化，同时我们还可以通过其自身的 CLEF（Clementine Extension Framework）框架和 Java 开发新的建模插件，其扩展性相当好，是一个不错的商业 BI 方案。

（3）Matlab

Matlab 也是一个商业软件，从名称上就可以看出它是为数学服务的。Matlab 的计算主要基于矩阵。其功能非常强大，涵盖了生物统计、信号处理、金融数据分析等一系列领域，是一个功能很强大的数学计算工具。

（4）R

R 是一个开源的分析软件，也是一个分析能力不亚于 SPSS 和 Matlab 等商业软件的轻量级（仅指其占用空间极小，功能却是重量级的）分析工具。R 支持 Windows、Linux 和 Mac OS 系统，对于用户来说使用起来非常方便。

R 和 Matlab 都是通过命令行来进行操作的，这一点很适合有编程背景或喜好的数据分析人员。R 的官方包中自带了相当丰富的分析命令和函数以及主要的作图工具。但 R 最大的优点在于其超强的扩展性，你可以通过下载扩展包来扩展其分析功能，并且这些扩展包也是开源的。R 社区拥有一群非常热心的志愿者，这使得 R 的分析功能一直都很丰富。

1.4 R 在数据分析中的优势

在数据分析中，重要的是要解决如何将数据分析技术，集成到复杂的业务信息应用环境中。对于数据科学家来说，R 语言无疑是他们的最佳选择。

R 作为一门编程语言在以下三个方面具有很强的优势：数据处理、统计和数据可视化。和其他数据分析工具不同的是，它是由统计学家开发的，是免费的软件，并且可以通过用户开发的包

进行扩展。目前在 CRAN 中大约有 4 800 多个包，而且包的数量一直是在不断增加的。

R 的好处不仅仅在于其是免费的，更重要的是其是开源的，所以它很灵活，更新速度也快，且集思广益。而且 R 有点像是一种网络，用的人越多，贡献的人也越多，这样其价值就成几何级数上升。

R 的编程思想非常简单，几乎就像写数学公式一样简单，学过 C 和 C++ 等底层语言的人就会知道 R 的编程是很简单的，且 R 是一种面向对象的高级语言。只要有人稍加指点，R 入门者很快就能学会其基本操作！

总之，R 的特点是功能强大、开源和免费，其在数据分析方面的主要优势如下：

- R 具有强大的数学统计分析功能，是可以使用的最为全面的统计分析包。它综合了所有标准的统计测试、模型和分析，同时还为管理和处理数据提供了一种全面的语言。统计方面最新的技术和创意，通常首先在 R 中出现。
- R 拥有 4800 多个可用的、高质量的、来自不同领域的软件包，这些软件包涵盖了从统计计算到机器学习，从金融分析到生物信息，从社会网络分析到自然语言处理，从各种数据库各种语言接口到高性能计算模型，可以说无所不包，无所不容，这也是为什么 R 正在获得越来越多各行各业的从业人员喜爱的一个重要原因。
- R 具有强大的科学数据可视化功能，能提供各种统计分析及图形显示工具，图形能力出色。它提供了一种完全可编程的图形语言，胜过了大多数其他统计和图形软件包。对数据分析来说，数据可视化是锦上添花的一部分，对于从数据中挖掘出规则和信息，将这些结果可视化出来，R 提供了优秀的的数据可视化功能。
- R 是开源软件，允许任何人对其修改。根据 GNU 组织的批准，其版权由做统计计算的 R 基金会拥有。R 欢迎任何人提供错误修复、代码改善和新的软件包，而且，对 R 来说，可用的大量高质量的软件包，是用这种方法来进行软件研发和共享的证明。
- R 是免费软件，没有特许限制，允许任何人来使用它，因此，我们可以在任何地点、任何时间运行它，甚至可以根据特许的条件而出售它。
- R 是一个支持跨平台的软件。R 可以在常见的操作系统和不同的硬件上运行，比如它可以在 GNU/Linux、Macintosh 以及 Microsoft Windows 上运行，既可在 32 位处理器上运行，也可在 64 位的处理器上运行。
- R 的扩展性很强，它能兼容、方便地导入其他很多不同格式的工具和输入数据，比如，来自 CSV 的文件、SAS、SPSS 以及 Matlab 的工具软件，或者是直接来自于 Excel、Access、Oracle、MySQL 以及 SQLite 的数据，R 都提供了交互的接口。R 还可以产生 PDF、JPG、PNG 和 SVG 格式的图片输出，以及用于 LATEX 和 HTML 的表格输出。

第 2 章

数据的读取与保存

作为一个资深的 NBA 詹姆斯粉丝，我能期望球场上的 LBJ 做各种事情，胜任各种位置；但我们不能期望一个软件可以做所有的事情，R 当然也不例外。在进行数据分析前，我们首先要获取原始数据，它们的形式可能是多种多样的，R 不一定能够直接读取，因此要想方设法将这些来源不同、形式不同的数据导入到 R 软件中。

本章我们将一一介绍各类数据的读取和存储方法，为读者提供一个尽量全面、丰富的参考。

2.1 数据读取

数据分析做到一定深度后，你会发现还是非常有趣的，但为数据分析读入数据以及把结果导出到其他系统以方便报表编写，可能是一件比数据分析更花时间和难办的事情。R 往往需要和其他软件工具协作，如 Excel、SPSS、网页，包括数据库系统。由于 R 不是专门做数据管理的工具，因而常需要专业数据库的辅助。不同类型的数据格式都需要 R 特殊处理、区别对待。还好，R 提供了强大的数据获取功能，R 自身以及从 CRAN 获得的程序包里面，具有丰富的数据导入和导出功能，

我们常说“万事开头难”，作为数据分析的第一步，读取数据也不是一件轻松的事，尤其当数据的来源、格式比较复杂时。保证数据完整而无遗漏地读入到 R 里面，后续的数据分析才有可能顺利展开。由于数据类型的多样化，R 读取这些数据的方法也不尽相同，下面让我们一起来逐个讨论。

2.1.1 读取内置数据集

R 本身提供了超过 50 个数据集，同时在功能包（包括标准功能包）中附带了更多的数据集。R 自身提供的数据集存放在自带的 `datasets` 程序包中。

通过指令 `data()` 可以列出基本系统提供的全部数据集（包括 `datasets` 以及通过 `library()` 加载的程序包中的数据集）。也可以载入特定的数据集：

```
> data() #查看数据集列表
> data(CO2) #载入 CO2 数据集（来自 datasets）
```

用 `library()` 载入其他程序包后，通过 `package` 参数查看包内附带的数据集。注意，用 `library()` 挂接 `package` 后，它的数据集也自动包含到搜索路径中了。

```
> library(MASS) #载入 package MASS
> data(package="MASS") #查看 MASS 中数据集
> data(SP500, package="MASS") #载入 MASS 中的 SP500 数据集，也可简化为 data(SP500)
```

2.1.2 读取文本文件

在 R 中读取文件，需要通过工作目录来完成。如果一个文件不在当前的工作目录中，则首先需要给出它的路径。

（1）文件目录操作

使用 `getwd()` 指令可以返回当前工作目录，`setwd()` 更改工作目录，但要注意的是，在 R 中文件路径的分隔符应为 “/” 或者 “\\”，不允许包括符号 “\”。更改目录也可以通过 R 菜单命令 `file->change dir...` 来完成。

```
> getwd() #返回当前工作目录
[1] "D:/R-3.0.1/bin/i386"
> setwd("d:/data") #也可以写成 setwd("d:\\data")
> getwd()
[1] "d:/data"
```

（2）常用的读取指令 read

R 最常用的读取文本文件（ASCII）的指令是 `read.table()`，它是读取矩形格子状数据最为便利的方式。使用 `read.table()` 指令读入后创建一个清单（list）。

`read.table()` 指令的格式如下：

```
read.table(file, header = FALSE, sep = "", quote = "\"", dec = ".", row.names, col.names,
  as.is = !stringsAsFactors, na.strings = "NA", colClasses = NA, nrows = -1,
  skip = 0, check.names = TRUE, fill = !blank.lines.skip, strip.white = FALSE,
  blank.lines.skip = TRUE, comment.char = "#")
```

`read.table()` 中包含很多参数，如表 2.1 所示，除前几个常用的参数外，其他参数一般情况下用默认值就可以了。

表 2.1 read.table()的参数设置

参数名	含 义
file	要读取的数据文件名称
header	逻辑值，TRUE 表示文件的第一行包含变量名，默认为 FALSE
sep	文件中字段的分隔符，默认为 sep=" "，表示分隔符是空格
quote	设置如何引用字符型变量。默认情况下，字符串可以被引号"或'括起，如果没有设定分隔字符，引号前面加\，即 quote=""
dec	设置用来表示小数点的字符
row.names	向量的行名，默认为 1,2,3,...
col.names	向量的列名，默认为 V1,V2,V3,...
na.strings	赋给缺失数据的值 (NA)
skip	开始读取数据前跳过的数据文件的行数
strip.white	是否消除空白字符
blank.lines.skip	是否跳过空白行

接下来用实际案例来说明，我们将要读入的数据文件是 15 个城市的工资指数，文件格式为 txt，存储在“d:\data”下。首先设定工作目录，再读入数据：

```
> setwd("d:/data")
> data=read.table("salary.txt",header=T)
```

如果没有事先设定工作目录，也可以直接读入完整的路径名称：

```
> data=read.table("d:/data/salary.txt",header=T)
> data
```

	CityWork	Price	Salary
1	Amsterdam	1714 65.6	49.0
2	Bombay	2052 30.3	5.3
3	Chicago	1924 73.9	61.9
4	Dublin	1759 76.0	41.4
5	Frankfurt	1650 74.5	60.4
...			

【注意】考虑到篇幅限制，仅显示部分运行结果。

与 read.table 用法类似的指令还包括 read.csv 和 read.delim：

- read.csv()用于读取逗号分隔文件，sep 默认值为","；
- read.delim()针对使用其他分隔符的数据（并且不使用行号），sep 默认值为"\t"。

这两个指令的其他参数设置与 read.table 的非常类似，但要特别注意 header 的默认值是 TRUE。

这两个函数对应的变形 read.csv2()和 read.delim2()是专门为以逗号为小数点的国家设计的，我们一般不用。

如果上面例子中的文件格式是 csv，则读入的命令改为：

```
> data=read.csv("salary.csv",header=T)
```

要注意的是，使用 read.table 或 read.csv 指令时，对数据格式的要求非常严格，数据必须是完整的，每一行数据的数量都一样。如果出现缺失值，用 read.table 读取时会报错，用 read.csv 读取时会自动在缺失位置填补 NA。

(3) 灵活的读取指令 scan()

read.table()对数据要求较高，并且它不是一种有效地读取大数据量矩阵的方法，下面要介绍一个更灵活的指令——函数 scan()。

```
scan(file = "", what = double(), nmax = -1, n = -1, sep = "",
      quote = if(identical(sep, "\n")) "" else "'\"'", dec = ".", skip = 0, nlines
      = 0, na.strings = "NA",
      flush = FALSE, fill = FALSE, strip.white = FALSE, quiet = FALSE,
      blank.lines.skip = TRUE, multi.line = TRUE, comment.char = "",)
```

scan函数的参数如表2.2所示，大部分参数与read.table的设置相同，需要注意的是，没有header这一参数，而且仍有几个需要加以特殊说明。scan()函数中如果不加参数，则可以手动输入数据。

表 2.2 scan()的参数设置

参 数	含 义
what	指定要读取的数据类型，支持的类型有 logical、integer、numeric、complex、character、raw 和 list，默认为数值型向量
nmax	指定要读入数据的最大数量，如果 what=list，nmax 则为可以读取的行数。默认情况下，将读取到文件末端
n	要读取数据的最大数量，默认值为没有限制

因为 what 参数可以灵活设置，故 scan()就可以创建不同类型的对象。在下面的例子中，what=list(City="",Work=0,Price=0,Salary=0)说明要创建列表，并指定了列表中对象的名称，这是一个名义列表结构，用来指定第一个变量 City 是字符型，后面三个是数值型变量。

```
> data2=scan("salary.txt",skip=1,what=list(City="",Work=0,Price=0,Salary=0))
#由于不存在 header 参数，skip=1 说明读取时跳过表示名称的第一行
Read 15 records
> data2
$City
[1] "Amsterdam" "Bombay"      "Chicago"      "Dublin"
[5] "Frankfurt"  "London"      "LosAngeles"   "Luxembourg"
[9] "MexicoCity" "NewYork"     "Paris"        "Singapore"
[13] "Sydney"     "Taipei"      "Tokyo"
$Work
[1] 1714 2052 1924 1759 1650 1737 2068 1768 1944 1942 1744
```

```
[12] 2042 1668 2145 1880
$Price
[1] 65.6 30.3 73.9 76.0 74.5 84.2 79.8 71.1 49.8
[10] 83.3 81.6 64.4 70.8 84.3 115.0
$Salary
[1] 49.0 5.3 61.9 41.4 60.4 46.2 65.2 71.1 5.7 65.8 45.9
[12] 16.1 52.1 34.5 68.0
```

读入数据后，可以通过一些简单的指令查看数据的基本信息。`mode()`指令用来显示对象的类型；`names()`显示对象中的标签，在上面的例子中就是 `list` 包含的变量名称；`dim()`显示对象的维数，如 `data` 文件中包含 15 个观察记录、4 个变量。

```
> mode(data) #显示对象的类型
[1] "list"
> names(data) #显示对象中的标签
[1] "City" "Work" "Price" "Salary"
> dim(data) #显示对象的维数
[1] 15 4
```

要显示列表中的变量，需要使用符号 `$`，但是当数据文件中有很多变量时，多次使用 `$` 会比较麻烦，这时用 `attach()` 指令，可以直接通过变量名称来获取变量中的信息。`detach()` 可撤销操作。

```
> data$Salary
[1] 49.0 5.3 61.9 41.4 60.4 46.2 65.2 71.1 5.7 65.8 45.9
[12] 16.1 52.1 34.5 68.0
> attach(data)
> Salary
[1] 49.0 5.3 61.9 41.4 60.4 46.2 65.2 71.1 5.7 65.8 45.9
[12] 16.1 52.1 34.5 68.0
> detach(data)
> Salary
Error: object 'Salary' not found
```

2.1.3 读取固定宽度格式的文件

有些数据文件格式非常规整，但没有分隔符，就需要我们在读取时手动划分每个字段的长度，这时需要用到的函数是 `read.fwf()`，它以行的方式首先读入数据，通过 `widths` 参数指定一个向量，来设置各个字段的宽度，注意小数点也占一个字符。

例如有一个 `txt` 文件（`d:\data\fwf.txt`），如下所示：

```
AA200530.31.2
AB200773.91.3
BB201184.21.4
BC201083.31.5
CC200981.61.6
```

通过如下命令将 txt 文件读入，widths 分别指定 4 个变量的宽度，col.names 指定 4 个变量的名称：

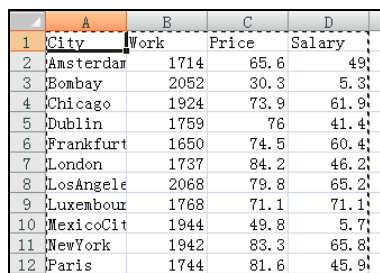
```
> data.fwf=read.fwf("d:/data/fwf.txt",widths=c(2,4,4,3),col.names=c("W","X","Y","Z"))
> data.fwf
  W    X    Y    Z
1 AA 2005 30.3 1.2
2 AB 2007 73.9 1.3
3 BB 2011 84.2 1.4
4 BC 2010 83.3 1.5
5 CC 2009 81.6 1.6
```

注意：文本文档中最后一行的回车符很重要，这是一个类似于停止符的标识，否则读入时会显示“最后一行不完整”的警告，但也不影响数据读入的效果。

2.1.4 读取 Excel 数据

Excel 数据是我们平时最常用的表格数据，一般比较简单的数据分析工作都可以在 Excel 中完成，因此很多数据都会保存为.xls 和.xlsx 文件。用 R 做数据分析时，就免不了要与 Excel 互动。

在 R 中打开 Excel 表格数据有多种方式，最简单的一种是从剪贴板中读取数据。首先打开 Excel 表格，选中需要的数据后复制（按快捷键 Ctrl+C），如图 2.1 所示，这时我们需要的数据就被存放在剪贴板了。



	A	B	C	D
1	City	Work	Price	Salary
2	Amsterdam	1714	65.6	49.8
3	Bombay	2052	30.3	5.3
4	Chicago	1924	73.9	61.9
5	Dublin	1759	76	41.4
6	Frankfurt	1650	74.5	60.4
7	London	1737	84.2	46.2
8	Los Angeles	2068	79.8	65.2
9	Luxembourg	1768	71.1	71.1
10	Mexico City	1944	49.8	5.7
11	New York	1942	83.3	65.8
12	Paris	1744	81.6	45.9

图 2.1 复制 Excel 中的数据

然后在 R 中输入以下指令：

```
> data.excel=read.delim("clipboard") #clipboard 即剪贴板
> mode(data.excel);dim(data.excel)
[1] "list"
[1] 15 4
```

读入的数据集 data.excel 与上文中通过 read.table() 读入的数据集 data 完全一样，都是列表，含有 4 个变量、15 个记录。

但是当数据量较大时，直接导入比使用剪贴板更加方便，因此这里介绍另一种方法——使用

程序包 RODBC（可以从 R-project 的官网下载）。当计算机连接到网络时，也可以通过如下命令直接安装程序包：

```
> install.packages("RODBC")
```

RODBC 提供了 R 和各类数据库的一个接口，通过它可以实现 R 和 Access、Excel、dBase 和 SQL Server 等多种软件的连接。其中获取 Excel 连接的函数是 `odbcConnectExcel()` 和 `odbcConnectExcel2007()`，分别用来读取 Excel 2003 版（扩展名为.xls）和 2007 版（.xlsx）数据。读入 Excel 数据的具体步骤如下：

```
> library(RODBC)
> channel=odbcConnectExcel2007("d:/data/Salary.xlsx") #获取 Excel 连接
> sqlTables(channel) #列出 Excel 中的表格
```

	TABLE_CAT	TABLE_SCHEM	TABLE_NAME	TABLE_TYPE	REMARKS
1	d:\data\Salary.xlsx	<NA>	Sheet1\$	SYSTEM TABLE	<NA>
2	d:\data\Salary.xlsx	<NA>	Sheet2\$	SYSTEM TABLE	<NA>
3	d:\data\Salary.xlsx	<NA>	Sheet3\$	SYSTEM TABLE	<NA>

获取 Sheet1 中的数据，可以使用如下任意一种方式。`sqlFetch()`直接读取 Excel 连接中的一个表到 R 数据框或列表中，`sqlQuery()`在 Excel 连接上执行 SQL 查询语句，并返回结果。

```
> data.excel2=sqlFetch(channel,"Sheet1")
> data.excel2=sqlQuery(channel,"select*from[Sheet1$]")
> close(channel) #关闭 ODBC 连接，释放空间
> mode(data.excel2);dim(data.excel2)
[1] "list"
[1] 15 4
```

2.1.5 读取数据库文件

R 软件一般将数据置于内存，数据处理能力有限。然而在当前这个“大数据”时代下，数据集的容量很大，往往以数据库文件的形式出现，这种情况下，我们使用 R 进行数据分析、数据挖掘前，都会先用 RODBC、RJDBC、RMySQL 等相关的包来调用数据库。然而，基本上各大数据库厂商已有相应的 R 语言企业级应用产品，这些厂商包括 Oracle、IBM、Teradata、Sybase、SAP 等。

- Oracle R Enterprise

Oracle R Enterprise 针对于大数据市场，用于处理日益丰富的数据，它将 R 的瓶颈完全打开。Oracle R Enterprise 允许 DBA 将 R 语言模型产品化，可以将 R 模型整合到 BI 仪表盘（BIEE），提供了高性能的代数运算（在 R 中整合 Intel's Math Kernel Library），高度整合了 R 语言快速开发、数据库并行计算的优势。

- IBM Netezza

Netezza 对 R 语言的支持，主要通过调用 R Enterprise from Revolution Analytics 平台来实现。Netezza 的特点可以总结为：可扩展、高性能、大规模内置并行分析平台。

- IBM InfoSphere BigInsights

IBM BigInsights 同样也整合了 R 语言资源, 提供了 MapReduce 架构的 R 语言并行化计算环境, 包括了大数据集的文本挖掘和机器学习算法。BigInsights 可以将构建的 R 语言模型发布在 Hadoop 平台上 (同 IBM Netezza 一样, 通过调用 R Enterprise from Revolution Analytics), 这可以极大地满足企业级数据需求。

- SAP HANA

借助 SAP Business Objects Predictive Analysis 平台, 分析师们既可以使用内置的预测性算法来构建模型, 也可以整合并使用流行的开源数据统计分析语言——R 语言。并且, 依托 SAP HANA 平台可以进行 in-database 分析。

- Teradata

Teradata 提供了免费的 teradataR 包, 用于在 R 环境下连接 Teradata 数据库、创建数据、调用 in-database 分析函数。

- Sybase RAP

Sybase RAP 主要用于对金融市场的实时分析, 其中 RAPStore 组件提供了内置分析函数, 包括时间序列分析函数、OLAP 函数、R 语言整合函数以及用户自定义函数, 其适用于大数据环境。同时, 还可以在 R 语言环境下通过 RJDBC 访问 Sybase RAP, 进行数据预处理, 避免在 R 中做数据清洗占用大量内存。

以上这些平台主要出现在企业中, 进行大数据的高端处理。接下来我们还是回到 R 软件中, 看看各程序包在 R 与数据库的连接中如何应用。

(1) 通过 RODBC 读取数据库

前面提到过, 程序包 RODBC 提供了 R 和各类数据库的连接。ODBC (Open Database Connectivity) 是 Microsoft 提出的数据库访问接口标准, 通过返回 ODBC 连接, RODBC 提供了从数据库中提取数据、回传数据等功能, 并且能够很方便地使用 SQL 语句。通过 odbcDataSources() 函数可查看可用的数据源:

```
> odbcDataSources()
      MS Access Database
"Microsoft Access Driver (*.mdb)"
      Excel Files
"Microsoft Excel Driver (*.xls)"
      dBASE Files
"Microsoft dBase Driver (*.dbf)"
```

程序包 RODBC 中最基础的函数为 odbcConnect(), 其可以直接返回一个 ODBC 连接。与 Excel 类似, 获取 Access 连接的函数分别为

```
odbcConnectAccess(access.file, uid = "", pwd = "", ...)
```

```
odbcConnectAccess2007(access.file, uid = "", pwd = "", ...)
```

其中，`access.file` 表示要读入的文件名称，`uid` 和 `pwd` 分别表示用户名和密码。读入 Access 数据的步骤与读入 Excel 的完全相同，因此不再举例说明。在 R 中获取数据库连接后，可以进行一系列 SQL 语句的操作，主要的函数如表 2.3 中所示。

表 2.3 RODBC 中与 SQL 相关的函数

函数名	功 能
<code>sqlFetch</code>	读取 ODBC 连接中的一个表到 R 的数据框中
<code>sqlQuery</code>	在 ODBC 连接上执行查询语句并返回结果
<code>sqlTables</code>	给出 ODBC 连接对应的数据库中的数据表
<code>sqlCopy</code>	复制 ODBC 连接中的查询结果到另一个 ODBC 连接中
<code>sqlDrop</code>	删除 ODBC 连接中的一个表
<code>sqlClear</code>	清空 ODBC 连接中的指定数据表内容

由于使用 RODBC 读取数据库的操作与读取 Excel 的方法基本上没有差别，故可参考上一节的案例，这里就不再赘述了。

Oracle 数据库是非常常用的数据库，它也可以通过 RODBC 进行读取，请看下面这个小例子。

假设 Oracle 的数据库名是 LEV2，用户名为 L2st，密码是 sa，并且在 LEV2 数据库中有一个表 TC_SNT，里面有若干字段，其中有 ID 和记录时间 RECORD_TIME。下面把一些常用的代码语句列出：

```
#加载库
library(RODBC)
#建立连接访问数据库
channel<-odbcConnect("LEV2",uid="L2st",pwd="sa")
#访问数据表 TC_SNT
snt1<-sqlFetch(channel,"TC_SNT")
#查询 ID=1 的历史值，按记录时间逆序排列
wa=sqlQuery(channel,'SELECT * FROM TC_SNT WHERE ID=1 order by RECORD_TIME desc ');
#提取最近 24 个数据
wa2=head(wa,n=24)
```

(2) 通过 RMySQL/DBI 读取数据库

程序包 RMySQL 提供了针对 MySQL 数据库系统的接口，早期版本存在不一样的接口，当前需要 DBI 程序包的支持，DBI 的主要功能是由字符向量产生一个合法的 SQL 标识，从而提供一个读取 SQL 的接口，MySQL 执行数据库接口 (DBI)。可以简单地理解为程序包 DBI 提供函数和接口，而 RMySQL 提供方法，是 R 读入 MySQL 数据库的一个驱动器，两个程序包合作完成对 MySQL 数据库的连接。

加载程序包后，就可以直接调用 DBI 中的函数，`dbDriver()` 会返回一个数据库连接管理对象，调用 `dbConnect()` 可打开一个数据库连接，而泛型函数 `dbDisconnect()` 用来关闭连接。特别地，对于 Oracle（甲骨文）和 SQLite 系统，分别使用程序包 `ROracle` 或 `RSQLite` 调用 `dbDriver()` 函数，而不是 `RMySQL`。

SQL 查询可以通过 `dbGetQuery` 或 `dbSendQuery` 传给数据库管理系统。`dbGetQuery` 传送查询语句，把结果以数据框形式返回。`dbSendQuery` 传送查询，还可以通过调用 `dbClearResult` 清除结果。

函数 `fetch()` 用于获得查询结果的部分或全部行，并以列表返回。函数 `dbHasCompleted()` 确定是否所有行都已经获得，而 `dbGetRowCount()` 返回结果中行的数目。

以上都是数据库连接、获得 SQL 查询的函数，当读入这些结果后，`dbReadTable()` 和 `dbWriteTable()` 实现 R 数据框的读入和将其存储至数据库，把数据框的行名映射到 MySQL 表的 `row_names` 字段。

目前 `RMySQL` 只支持在 Linux 和 Macintosh 环境下运行的包，若在 Windows 环境下使用 `install.packages("RMySQL", type="source")`，则得到的结果是下载正确，但安装失败。若要在 Windows 环境下安装，首先要安装 MySQL 数据库，官网在 `RMySQL` 介绍页面给出的 URL 链接为 <http://biostat.mc.vanderbilt.edu/wiki/Main/RMySQL>，按步骤进行安装。

假设安装 MySQL 时，设置 `user` 为 `root`，密码是 6 个 1，并且在 MySQL 中已经建立了名为 `test` 的数据库，里面有一个表 `students`，其中有三个字段 `name`、`age` 和 `sex`。下面把一些常用的代码语句列出：

```
> library(RMySQL) #加载 RMySQL 包，同时也会加载 DBI 包
# 打开一个 MySQL 数据库的连接
> con=dbConnect(MySQL(),user="root",password="111111",dbname = "test")
# 将数据库中的表名存入 table.names，方便查看
> table.names=dbListTables(con)
# 列出表 students 中的字段
> field.names=dbListFields(con,"students")
#获得并列出整个表
> dbReadTable(con,"students")
> dbSendQuery(con, "SET NAMES gbk") #传送查询，说明用什么字符集来获取数据库字段，gbk 或 utf8
    要与之前设置的保持一致。
> query=dbSendQuery(con, "select * from students order by age")
> fetch(query) #显示以年龄排序的查询结果
# 删除表（删除成功后显示逻辑值 TRUE）
> dbRemoveTable(con,"students")
# 关闭连接
```

```
>dbDisconnect(con)
```

(3) 通过 RJDBC 读取数据库

R 连接数据库实现的技术主要包括 ODBC 和 JDBC 两大方面，JDBC (Java Data Base Connectivity) 是 Java 数据库连接，其由一组用 Java 语言编写的类和接口组成。

程序包 RJDBC 提供了基于 JDBC 接口的数据库连接功能，同时需要程序包 rJava 的支持。和 RMySQL 一样，RJDBC 仍然调用 DBI 中的函数实现数据库连接，其也是一个驱动器的功能，不同的只是数据库接口变为 JDBC。RJDBC 支持 Windows 版本，在安装的同时也将 rJava 一同安装进 R。

首先通过函数 JDBC() 创建一个新的 DBI 驱动程序，用于启动 JDBC 连接。函数 JDBC() 的常用格式为：

```
JDBC (driverClass = "", classPath = "", identifier.quote = NA)
```

其中，driverClass 指定要加载的 JDBC 驱动程序的 Java 类名；classPath 指定所需的 JDBC 驱动程序的类路径，通常是包含驱动程序的 JAR 文件的路径；identifier.quote 指定引号标识符。

通过 JDBC 获取连接后，读入数据库和 SQL 查询的函数均调用自程序包 DBI，语句与上一节介绍的完全一致。

```
>library(RJDBC)
>help(JDBC)
>drv=JDBC("com.mysql.jdbc.Driver", "/etc/jdbc/mysql-connector-java-3.1.14-bin.jar", "`")
>conn=dbConnect(drv, "jdbc:mysql://localhost/test")
>dbListTables(conn) #列出数据库中的表
>dbGetQuery(conn, "select count(*) from iris") #执行查询
```

如果需要进一步深入地了解，可以使用 help(JDBC)，查看程序包中自带的例子。

2.1.6 读取网页数据

可扩展标记语言 (Extensible Markup Language, XML) 用于标记电子文件，使其成为结构性的标记语言，可以用来标记数据、定义数据类型，是一种允许用户对自己的标记语言进行定义的源语言。

在实际的数据分析中，最常见的 XML 语言就是网页数据，我们经常要从网页获取表格数据，例如股票、债券数据往往需要从网站上直接获得。

R 中的程序包 XML 为读写 XML 文档提供了通用的工具。

在 R 中要读取网页上的 HTML 表格数据，要利用程序包 XML 中的 readHTMLTable() 函数，其调用格式如下：

```
readHTMLTable(doc, header = NA, colClasses = NULL, skip.rows = integer(), trim = TRUE,
               elFun = xmlValue, as.data.frame = TRUE, which = integer(), ...)
```

其主要参数如表 2.4 所示。

表 2.4 readHTMLTable()的参数设置

参数名	含 义
doc	HTML 文件或 URL（网页网址）
header	若为逻辑值，表示是否包含列标签；若为字符向量，则为列名称赋值
colClasses	一个列表或向量，指定表中的各列数据的类型。除了通常的"integer"、"numeric"、"logical"和"character"，还可以使用 FormattedNumber 引入一个新的类
skip.rows	指定要忽略的行
trim	逻辑值，表示是否要删除开头和结尾的空白单元格
which	整数向量，表示返回网页中的哪几个表格

下面以东方财富网站的股票数据为例，我们使用 R 提取网页上的股票数据。

首先提取网址，通过 readHTMLTable 读取表格，一个网站页面可能包含多个表格，使用 which=1 指定要读取的是第一个表。从结果可以看出，我们读取的数据 table 是一个列表，有 4 行和 7 个变量。

```
> install.packages("XML") #安装解析 XML 的包
> library(XML)
> baseURL="http://data.eastmoney.com/center/stock.html" #存入网址
> table=readHTMLTable(baseURL,header=TRUE,which=1)
> mode(table);dim(table) #查看 table 的类型和数据维度
[1] "list"
[1] 4 7
```

注意：使用 XML 程序包时，由于 XML 包编码方式的问题，若表格的变量名为中文，则读入后会出现乱码，如输入：

```
> head(table,2) #查看列表 table 前两行的数据
```

从结果可以看出，显示中有乱码。因此，需要对表格中的变量名重新赋值。若 names 为英文则不会出现问题。

```
> names(table)=c("类别","成交量(万股)","成交金额(亿元)","总市值(亿元)","流通市值(亿元)","
上市公司(家)","平均市盈率") #给变量名重新赋值
> table$类别=c("沪市","深市","中小板","创业板") #给第一个变量"类别"重新赋值
> head(table,2)
```

替换中文字符后，没有乱码了。

笔者作为一个 NBA 的球迷，詹姆斯的铁杆粉丝，常常在篮球网站上浏览球员的职业生涯数

据，这些数据都是以 HTML 表格出现的。在一些评论文章中，作者会对数据进行绘图、比较，以得到更有说服力的球员对比情况，其实这些我们也可以在 R 中实现，第一步就是先把网页上的表格导入。

例如读取勒布朗·詹姆斯的 10 年生涯数据，会得到一个有 26 个变量的列表，该列表记录了他每年的平均得分、篮板和命中率等信息，这些都是以英文和数字记录的，所以读入后不会出现乱码。

```
> u="http://www.basketball-reference.com/players/j/jamesle01.html"
> James <- readHTMLTable(readLines(u), which=3, header=TRUE)
> dim(James)
[1] 10 26
> James[1:5,1:10] #查看前 5 年的数据，取其中前 10 个变量
  Season Age  Tm  Lg Pos  G  GS  MP  FG  FGA
1 2003-04  19  CLE NBA  SG  79  79 3122 622 1492
2 2004-05  20  CLE NBA  SF  80  80 3388 795 1684
3 2005-06  21  CLE NBA  SF  79  79 3361 875 1823
4 2006-07  22  CLE NBA  SF  78  78 3190 772 1621
5 2007-08  23  CLE NBA  SF  75  74 3027 794 1642
```

2.1.7 读入 R 格式的文件

R 的数据或更一般的对象（如向量、矩阵、数据框、列表甚至函数等）可以通过 `save()` 保存为 R 专有的文件格式，以 `Rdata` 为后缀。要读取这类文件，需要用到函数 `load()` 来加载。

例如先前把数据集 `data` 的数据存储在 `salary.Rdata` 中，注意数据框的名称与文件名不一致，可通过如下命令重新载入数据框 `data`：

```
> load("d:/data/salary.Rdata")
> head(data,5) #显示数据框前 5 行的记录
  City Work Price Salary
1  Amsterdam    1714  65.6  49.0
2  Bombay      2052  30.3   5.3
3  Chicago     1924  73.9  61.9
4  Dublin      1759  76.0  41.4
5  Frankfurt   1650  74.5  60.4
```

2.1.8 从其他统计软件读入数据

有时 R 需要读取其他统计软件的数据文件，如 SAS、SPSS、Stata、Minitab 和 S-PLUS 等，使用程序包 `foreign` 可以解决这一问题。

`foreign` 中包含读入各种统计软件数据的函数，如表 2.5 所示。

表 2.5 程序包 foreign 的主要函数

函 数	功 能
read.spss()	读取 SPSS 里面 save 和 export 命令创建的文件，返回一个列表含有标签 (label) 的 SPSS 变量，可以选择转换为 R 中的因子 (factor)
read.mtp()	读入 Minitab 表格
read.ssd()	读入 SAS 永久数据集 (.ssd 或 .sas7bdat)
read.xport()	读入 SAS 传输格式 (XPORT) 的文件，并且返回一个数据框
read.S()	读取二进制数据文件，或由 S-PLUS 版本 3 以上产生的数据转存文件。它能读取的不是全部 S 对象，只能读入向量、矩阵、数据框和列表
read.dta()	读入 Stata 版本 5~11 的二进制文件。有标签的 Stata 变量同样可以选择转换为因子
read.epiinfo()	读取 Epi Info 版本 6 及更早的 REC 格式的数据文件。Epi Info 是由美国疾病控制中心提供的免费数据录入和管理软件
read.systat()	读取 Systat 在小字节序机器 (little-endian machines) (比如 Windows) 上保存的矩形的数据文件，这些文件的扩展名为 .sys 或 .syd

SPSS 和 SAS 是常用的两个统计软件，下面就以这两种软件的文件读入为例，介绍 R 加载其他类型文件的方法。

(1) 读取 SPSS 数据

SPSS 数据存放在扩展名为 .sav 的文件中，通过指令 read.spss() 读取。

```
> library(foreign)
> data.spss=read.spss("d:/data/salary.sav",to.data.frame=T) #data.spss 读入后为数据框变量
Warning message:
In read.spss("d:/data/salary.sav", to.data.frame = T) :
  d:/data/salary.sav: Unrecognized record type 7, subtype 18 encountered in system file
> dim(data.spss)
[1] 15 4
```

输入语句后会出现警告，这可能是由于 SPSS 文件包含很多附加信息，如变量类型、变量长度等，R 在读入时无法全部识别才会出现警告信息，但这并不影响完整地读入数据框 data.spss，工资的记录都完整保存了下来，仍然包含 15 条记录，4 个变量。

对于 SPSS 数据的读取，还可以使用程序包 Hmisc 中的函数 spss.get()，它可以导入更多的附加信息，例如变量的标签 (label)。

```
> library(Hmisc)
> data.spss2= spss.get("d:/data/salary.sav")
```

(2) 读取 SAS 数据

相比 SPSS，SAS 文件的读取更为复杂一些，因为 SAS 软件是一款可以处理大数据集的编程统计软件，其本身的复杂程度就高于 R 和 SPSS，它的数据文件固定地存放在永久或临时的数据集中。

但由于 SAS 应用广泛，实际操作中很可能需要在 R 中调用 SAS 数据集。这里介绍一种常用的也是相对简便的方法，首先在 SAS 中生成传送文件，再到 R 中读取。

例如，要读取永久数据集 `sasuser` 中的公司员工信息，存放至 `company` 文件中，首先 SAS 生成传送文件的指令如下：在永久数据集 `r_sas` 下建立传送文件，存放至 ‘`d:/data`’ 路径下；打开传送文件 `company`；将永久数据集 `sasuser` 中文件 `company` 的数据全部复制到传送文件中去。

```
libname r_sas xport 'd:/data/company.xpt';
data r_sas.company;
set sasuser.company;
run;
```

SAS 中运行的结果是：

```
1 libname r_sas xport 'd:/data/company.xpt';
```

NOTE: 已成功分配逻辑库引用名 R_SAS, 如下所示:

引擎: XPORT

物理名: d:\data\company.xpt

```
2 data r_sas.Company;
3 set sasuser.company;
4 run;
```

NOTE: 从数据集 SASUSER.COMPANY 读取了 8 个观测值。

NOTE: 数据集 R_SAS.COMPANY 有 8 个观测值和 4 个变量。

NOTE: "DATA 语句"所用时间 (总处理时间):

实际时间 0.10 秒

CPU 时间 0.07 秒

接下来在 R 中读入传送文件就比较容易了，传送文件的扩展名为 `.xpt`，使用上面介绍的函数 `read.xport()` 即可。读取后生成一个数据框 `company`，从结果可以看出是一个具有 4 个变量、8 个观测值的数据框，这与上面的 SAS 运行结果一致。

```
> library(foreign)
> company=read.xport("d:/data/company.xpt")
> company
```

	name	age	sex	ssn
1	Morrison, Michael	32	M	
2	Rudelich, Herbert	39	M	029-46-9261
3	Vincent, Martina	34	F	074-53-9892

4	Benito, Gisela	32	F	228-88-9649
5	Sirignano, Emily	12	F	442-21-8075
6	Harbinger, Nicholas	36	M	446-93-2122
7	Phillipon, Marie-Odile	28	F	776-84-5391
8	Gunter, Thomas	27	M	929-75-0218

使用函数 `sas.xport.get()` 读取传送文件也可以得到相同的效果，但使用前必须同时加载 `Hmisc` 和 `foreign` 两个程序包。

```
> library(Hmisc)
> library(foreign)
> company1=sasxport.get("d:/data/company.xpt")
```

读取其他统计软件数据时，程序包 `foreign` 使用得最为普遍，代码编写也比较容易，但由于不同软件的特性，往往需要具体问题具体分析。因此，在数据读入的部分，`R` 与其他统计软件互相读取是最大的难点，实际工作中如果能够获得文本文档形式的原始数据，将大大简化读取工作的烦琐程度。

2.2 数据保存

前面花了大量篇幅介绍数据的导入，这对数据分析固然十分重要，但分析的一些中间过程或结果会产生新的数据集，我们也需要保存起来。根据需要，`R` 中的数据也可以保存为不同格式的文件。

2.2.1 使用函数 `cat()`

函数 `cat()` 是导出数据的基础，它除了可以在屏幕上输出对象之外，也能够输出成文件，其调用格式如下：

```
cat(... , file = "", sep = " ", fill = FALSE, labels = NULL, append = FALSE)
```

其中的参数 `file` 表示要输出的文件名，当参数 `append = TRUE` 时，在指定文件的末尾添加内容。

`R` 可以通过连续地调用 `cat` 对一个文本文件进行写入；但最好的方式是，特别是需要多次这样做的时候，首先为写入或添加文本打开一个 `file` 连接，然后用 `cat` 连接，最后关掉（`close`）它。

创建一个连接后默认是不打开的，泛型函数 `open` 和 `close` 可用于显式地打开或关闭连接，`file` 函数也可以用于打开链接。

首先举例说明 `cat()` 如何在屏幕上输出对象，它可以连接多个字符串，也可以连接字符串和数值向量等不同类型的对象。

```
> cat(c("AB", "C"), c("E", "F"), "n", sep="")
ABCEFn
> i=1:5
```

```
> cat("i = ", i, "n", sep=",") #以逗号为分隔符
i = ,1,2,3,4,5,n
```

这里更重要的功能是向一个指定的文件写入数据,例如要向 `cat.txt` 中存储数据,参数 `file` 指定被写入的文件,如果指定的文件已经存在则原来内容将被覆盖。若不想被覆盖,则要设置 `append=TRUE`,表示追加内容。使用 `readLines()` 函数可以直接从连接中以行的形式读取文本。

```
> cat(c("AB", "C"), c("E", "F"), file="d:/data/cat.txt", sep=".")
> readLines("d:/data/cat.txt")
[1] "AB.C.E.F"
> i=1:5
> cat(i, file="d:/data/cat.txt", append=TRUE)
> readLines("d:/data/cat.txt")
[1] "AB.C.E.F1 2 3 4 5"
```

要用 `cat()` 多次写入时,也可以通过 `file` 事先打开一个连接,以简化代码。

```
> a=file("d:/data/cat.txt")
> cat("1 2 3 4 ", "2 3 5 7", "11 13 15 17", file=a, sep="\n") #分隔符 sep="\n"表示换行
> read.table(a)
  V1 V2 V3 V4
1  1  2  3  4
2  2  3  5  7
3 11 13 15 17
```

2.2.2 保存为文本文件

通常我们会把 R 中的向量、矩阵、数据框和列表等对象写入一个文本文件中并保存起来,例如扩展名为 `.txt` 的文件,因为一般文本文档可以被各种软件读取,具有很强的普适性。

最常见的工作是把一个矩阵或数据框以数字的矩形网格方式写入文件中,而且还可能保留行列的标签。这可以通过函数 `write.table` 和 `write` 来完成。

函数 `write` 仅可以写出一个矩阵或向量的特定列(和对一个矩阵进行转置)。

函数 `write.table()` 则更为便利,它可把一个数据框或列表等对象以包含行列标签的方式写出。

函数 `write.table()` 的调用格式如下所示:

```
write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ", eol = "\n", na = "NA",
           dec = ".", row.names = TRUE, col.names = TRUE)
```

其中, `x` 表示要写入的对象,最好是矩阵或数据框; `quote` 是逻辑值, `TRUE` 表示变量名等字符、因子要用双引号括起来; `sep` 指定分隔符; `row.names/col.names` 也是逻辑值, `TRUE` 表示将行名/列名写入文件中。

例如,在数据读取中得到的数据框为 `data`,我们把它保存为简单的文本文件 `salary1.txt`。

```
> data=read.table("d:/data/salary.txt", header=T)
```

```
>write.table(data,file="d:/data/salary1.txt",col.names=T,quote=F)
```

可见, `read.table()` 和 `write.table()` 就像两个互逆函数一样, 保存后的 `salary1.txt` 和之前的 `salary.txt` 文件内容是一模一样的。

还可以使用 `write.csv()` 将数据框保存成逗号分隔文件, 方法与上面一样, 但是不包含列名, 即 `col.names = NA`。将 `row.names` 设置为 `FALSE`, 否则存入文件时会把行名 1,2,3,... 也写入。这样当再次读入 `csv` 文件时, 得到的数据框与 `data` 一样。

```
> write.csv(data,file="d:/data/salary1.csv",row.names=F,quote=F)
> data.csv=read.csv("d:/data/salary1.csv")
> dim(data.csv)
[1] 15 4
```

2.2.3 保存 R 格式文件

涉及多个数据集的统计分析经常使用 `Rdata` 文件存储和加载数据, 上文已经提到存储 `R` 格式文件使用 `save()` 命令。例如 `R` 中已经有数据框 `data`, 把它保存到 `Rdata` 文件中, 下次使用时可以用 `load()` 再加载。

```
> save(data,file="d:/data/salary1.Rdata")
>load("d:/data/salary.Rdata")
```

2.2.4 保存为其他类型文件

程序包 `foreign` 除了有用于读取其他统计软件的文件函数外, 还有用于存储的函数。主要的函数是 `write.foreign()`, 目前支持导出到 `SPSS`、`Stata` 和 `SAS`。

```
write.foreign(df, datafile, codefile, package = c("SPSS", "Stata", "SAS"), ...)
```

其中, `df` 是一个数据框, `datafile` 是输出数据的文件名称, `codefile` 是用于代码输出的文件名称等。

例如将数据框存为 `SPSS` 可读取的文件:

```
> library(foreign)
>write.foreign(data,datafile="d:/data/salary.sav",codefile="d:/data/code.txt",package="SPSS")
```

参数 `codefile` 没有默认值, 因此必须指定一个文件将代码存入。

运行后, 在 `d:/data/` 下会生成两个文件: 一个是 `SPSS` 可读取的文件, 一个是 `.txt` 文件。

存储的文件在 `SPSS` 中打开时并非直接可以读入, 还需要根据向导, 进一步设置分隔符、首行等。

另外, `foreign` 包中还有一个函数 `write.dta()`, 其可生成 `Stata` 二进制格式的文件; 以及 `write.dbf()` 可生成 `DBF` 文件, 它是 `dBase` 和 `FoxPro` 所使用的数据库格式。只是这两个函数并不常用。

第 3 章

数据预处理

数据是分析的核心，在做数据分析之前，首先要对数据进行一定的处理。数据预处理指当录入或读取数据后，对数据进行必要的清理，包括查错纠错、异常观察值和无效样本的处理、转换、填补缺失值等，这是数据分析的重要前提，是描述统计、定性定量分析的基础。它的主要目的就是为后续的分析工作提供经过清理、质量较好的数据集。

本章将讲述在 R 软件中如何对原始数据进行预处理，主要内容包括数据处理的基本函数、数据整理、数据修改与选择、缺失值处理等。

3.1 基本函数

并非所有的数据都是可以直接使用的，现实中收集的数据往往数量较大，信息又多又繁杂，因此很多数据都是参差不齐的，存在层次不清、数量级不同等问题，直接使用会给后续的数据分析和挖掘带来许多不便之处，甚至导致错误的结论。

例如保险公司的保费收入和赔付支出数据，由于每季度都收集，常常出现日期格式不统一的情况，而且数据缺失的情况也很常见。如果把数据分析过程比喻为下厨做菜，那么获取食材等同于读取数据，而洗菜、择菜等准备工作就类似于我们所说的数据预处理。

就像数据预处理是分析的预备工作一样，了解基本函数也是一个准备工作。预处理还是一个比较烦琐的过程，首先应该了解用 R 中的哪些工具、函数可以帮助完成这一工作。导入大规模的数据后，为了首先对数据集有一个大体的了解，通常需要从中提炼一些关键内容，例如均值、极差、分位数等。

R 中用来处理数据的函数非常非常多，而且使用起来非常简单。下面列出使用最频繁的函数，如表 3.1 所示。

表 3.1 基本数学函数

函 数	功 能
sum(x)	x 中元素的求和
prod(x)	x 中元素的连乘积
max(x) / min(x)	x 中元素的最大值/最小值
which.max(x) / which.min(x)	返回 x 中最大/最小元素的下标
range(x)	返回最值，相当于 c(min(x), max(x))
length(x)	x 的长度（元素个数）
mean(x)	x 中元素的均值
median(x)	x 中元素的中位数
quantile(x)	x 的 5 个分位数（0%、25%、50%、75%、100%）
summary(x)	返回 x 的 5 个分位数和均值
var(x)	x 中元素的方差（总体方差，分母用 $n-1$ 计算）；若 x 是一个矩阵或数据框，则计算协方差阵，这时等同于 cov(x)
sd(x)	x 中元素的标准差
cov(x,y)	x 和 y 的协方差；若是矩阵或数据框，则计算 x 和 y 对应列的协方差
cor(x)	x 是一个矩阵或数据框，计算相关系数矩阵；若 x 是一个向量，则结果为 1
cor(x,y)	x 和 y 的线性相关系数；若对象为矩阵或数据框，则计算相关系数矩阵

以上是最常用于统计描述的函数，比较基本，而表 3.2 中的函数可以返回更为复杂的结果，提炼数据中的更多信息。

表 3.2 高级数学函数

函 数	功 能
pmin(x,y,...) / pmax(x,y,...)	返回一个向量，其第 i 个元素是 x[i],y[i],...的最小值/最大值
cumsum(x)	返回一个向量，其第 i 个元素是 x[1],...x[i]的求和
cumprod(x)	同上，求乘积
cummin(x) / cummax(x)	同上，求最小/最大值
mad(x)	离差
round(x,n)	对 x 中的元素四舍五入，至小数点后第 n 位
sort(x) / order(x)	排序，默认升序（下文将详细介绍）
rank(x)	x 中元素的秩
rev(x)	对 x 中的元素取逆序
scale(x)	x 是一个矩阵，中心化和标准化数据
unique(x)	x 是向量或数据框，对重复的元素只取一个
na.omit(x)	忽略有缺失值 NA 的数据（x 是矩阵或数据框，则忽略行）
na.fail(x)	若包含缺失值，则返回错误消息（Error）
table(x)	统计 x 中完全相同的数据个数

续表

函 数	功 能
table(x,y)	x 和 y 的列联表
sample(x,size=n)	从 x 中抽取样本量为 n 的样本
skewness(x)	求偏度（属于程序包 timeDate）
kurtosis(x)	求峰度（属于程序包 timeDate）
emm(x, order)	k 阶原点矩（属于程序包 Actuar）

接下来以上一章读取的 salary 数据集为例，应用上面的函数对数据做一些基本处理，这里只为说明函数的使用方法，更加详细的描述性统计将在第 5 章具体介绍。

```
> data=read.table("d:/data/salary.txt",header=T)
> attach(data)
> mean(Salary) #求均值
[1] 45.90667
> length(Salary) #数据长度（个数）
[1] 15
> cumsum(Salary) #累积工资
[1] 49.0 54.3 116.2 157.6 218.0 264.2 329.4 400.5 406.2 472.0 517.9 534.0 586.1
[14] 620.6 688.6
```

可以看到，工资数据一共有 15 个记录，其均值为 45.9，cumsum()可以由左向右计算累积工资额。

但是像这种数据几乎都不相等的情况，尤其当数据量较多时，要想统计数值大小等基本信息比较困难，所以需要对其分组，从而大致描述数据信息。需要利用函数 cut()，它可以把数值型对象分区间转换为因子，调用格式如下：

```
cut(x, breaks, labels = NULL,include.lowest = FALSE, right = TRUE...)
```

其中，x 为被转换的对象，是一个数值向量；breaks 可以是单个数字，指明 x 要分为几组，也可以是一个向量，可自行设置分组的切点；labels 给每个组添加标签；include.lowest 是逻辑值，指明区间的开闭情况，即区间端点值是否包括在内；right 也是逻辑值，默认区间为左开右闭。

例如将 salary 划分为 3 组，用函数 table()统计数据信息，若没有设置标签，将直接显示各区间的端点数值：

```
> salary1=cut(Salary,3)
> table(salary1)
salary1
(5.23,27.2] (27.2,49.2] (49.2,71.2]
      3         5         7
> salary1=cut(Salary,3,labels=c("low","medium","high")) #给每个区间设置标签
> table(salary1)
```

```
salary1
  low medium high
3    5    7
```

落入低、中、高三个区间的工资个数分别是 7、5、3，由此我们大致对数值的分布有了一个了解。可见，当 `breaks` 为单个数值时，分组后区间的长度都是相等的，如果要自定义区间长度，则对 `breaks` 赋值一个数值向量，这样就可以清晰地看到落入不同工资段内的数据个数了。

```
> breakpoints=c(0,30,40,50,60,70)
> salary2=cut(Salary,breaks=breakpoints)
> table(salary2)
salary2
(0,30] (30,40] (40,50] (50,60] (60,70)
  3      1      4      1      5
```

当然对于数据的直观分析，画图是最便捷的方式，R 提供的绘图函数有许多，下一章将详细介绍。这里把常用的几个汇总在一个函数中，直接调用它，就可以对数据绘制出多个图形来，由此可以更直观地观察数据分布信息，如图 3.1 所示。

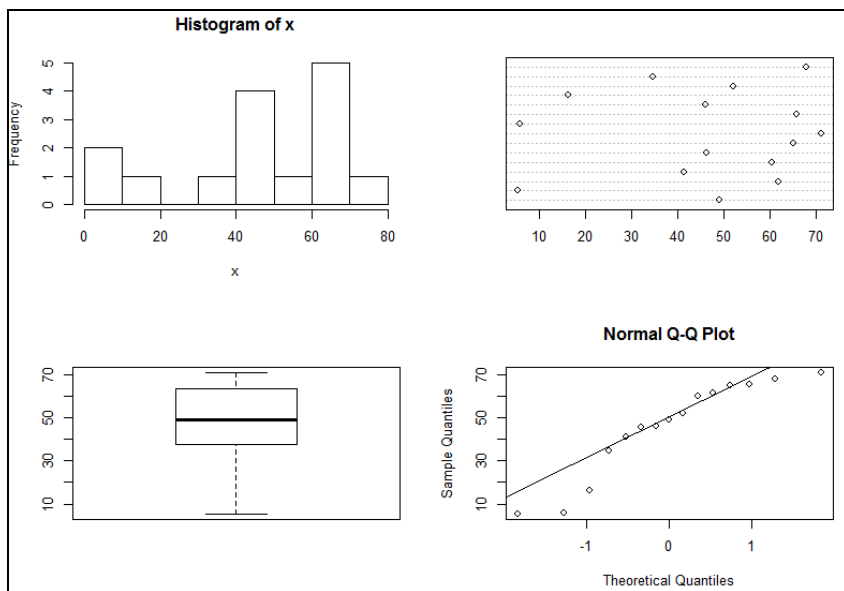


图 3.1 salary 数据的图形化显示

```
> pic=function(x){
+   par(mfrow=c(2,2))      #绘图区域分割为四部分
+   hist(x)                #直方图
+   dotchart(x)            #点图
+   boxplot(x)             #箱线图
+   qqnorm(x);qqline(x)    #正态概率图
```

```
+ par(mfrow=c(1,1)) #恢复单图区域
+ }
> pic(Salary) #调用编写好的函数pic()
```

3.2 数据修改

首先，进入 R 中的原始数据可能是五花八门的，就像插花一样需要修剪枝桠才能变得完美，数据也要经过整理才适用于各个统计分析方法。

数据修改和整理工作，有时被形容为对数据的“整形”，这不仅是为了改善数据的外观，也是统计分析和作图前的必要步骤。

3.2.1 修改数据标签

数据修改主要是细节上的工作，这里使用上一章的文本文件 salary，读取后存入数据集 data，下面说明如何在 R 中修改已读入的数据。

首先数据的标签是可以改变的，只需要使用上面提到的 names 函数就可以实现，如以下指令将 data 中原来的标签都变成大写：

```
> data=read.table("d:/data/salary.txt",header=T, stringsAsFactors =F)
> names(data)=c("CITY","WORK","PRICE","SALARY")
> names(data)
[1] "CITY" "WORK" "PRICE" "SALARY"
```

3.2.2 行列删除

R 主要使用[]来提取数据的子集，下文我们将具体讲到，这里同样使用中括号来删除数据的某行或某列，只需在行号或列号的前面加负号“-”即可，例如 data[-1,-3]表示删除数据集 data 的第一行和第三列（即 PRICE）：

```
> data2=data[-1,-3]
> data2
      CITY WORK SALARY
2    Bombay 2052   5.3
3   Chicago 1924  61.9
4    Dublin 1759  41.4
5  Frankfurt 1650  60.4
6    London 1737  46.2
...
```

3.3 缺失值处理

与数据清洗相比，以上的修改只是“小巫见大巫”。在收集的真实数据中，数据缺失是非常

常见的现象，而且影响重大，缺失值的处理可以说是数据预处理的重中之重。

一般而言，数据缺失主要是由以下几个原因造成的：

- 在数据收集阶段，某些记录或字段丢失；
- 调查访问中，被访者拒绝透露相关信息，导致数据的有效性；
- 由于机械原因，导致数据存储的失败。

基本上，缺失数据处理的流程是，首先判断其模式是否随机，然后找出缺失的原因，最后对缺失值进行处理。

3.3.1 判断缺失数据

R 中缺失值以 NA 表示，判断数据是否存在缺失值的函数有两个，最基本的函数是 `is.na()`，它可以应用于向量、数据框等多种对象，返回逻辑值。

还以上面数据为例，先将 `data` 文件中工资指数大于 65 的值替换为缺失值：

```
> attach(data)
The following object is masked from data (position 3):
CITY, PRICE, SALARY, WORK
> data$SALARY=replace(SALARY,SALARY>65,NA)
> is.na(SALARY)
 [1] FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE FALSE TRUE FALSE FALSE FALSE
[14] FALSE TRUE
> sum(is.na(SALARY))
[1] 4
```

`is.na()` 作用于向量 `SALARY` 后，如果对应的数值为缺失值则返回 `TRUE`，否则为 `FALSE`，使用求和函数 `sum()` 可以知道一共有 4 个缺失值。

另一个判断缺失值的函数是 `complete.cases()`，它同样返回逻辑值向量，但值与 `is.na()` 的相反：缺失值为 `FALSE`，正常数据为 `TRUE`，利用它来选取无缺失数据的行非常方便。

```
> complete.cases(data$SALARY)
 [1] TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE TRUE FALSE TRUE TRUE TRUE
[14] TRUE FALSE
```

3.3.2 判断缺失模式

存在缺失数据时，需要进一步判断数据的缺失模式，判断是否是随机的，然后才能确定处理的方法。

在处理缺失数据之前，首先来了解下程序包 `mice`，它的全称为 `Multivariate Imputation by Chained Equations`，即利用链式方程进行多元插补，可以处理混合变量类型的数据缺失，自动产生填补变量的预测变量，是处理缺失值的重要工具。

`mice` 包中的函数 `md.pattern()` 用于显示缺失数据的模式，其只有一个参数，就是要判断的矩阵或数据框。

为了说明更复杂的情况，我们将数据集中变量 `PRICE` 大于 80 的值也替换为缺失值。

```
> data$PRICE=replace(PRICE, PRICE>80, NA)
> install.packages("mice")
> library(mice)
> md.pattern(data)
  CITY WORK SALARY PRICE
8   1   1     1     1   0
3   1   1     1     0   1
2   1   1     0     1   1
2   1   1     0     0   2
    0   0     4     5   9
```

输出结果中的“1”表示没有缺失数据，“0”表示存在缺失数据。第 1 列第 1 行的“8”表示有 8 个样本是完整的，下面的“3”表示有 3 个样本缺少了 `SALARY` 这一变量的值，第 1 列最后一个数字“2”表示有两条记录在 `SALARY` 和 `PRICE` 上都有缺失。最后一行表示各个变量缺失的样本数合计。

`md.pattern()` 是从数值的角度判断缺失模式，另一种比较直观的方法是以图形方式描述缺失数据，需要用到程序包 `VIM` 的 `aggr()` 函数。

程序包 `VIM` 提供了在 R 中探索数据缺失情况的新工具，实现缺失模式的可视化，其可以帮助我们了解缺失值或数据错误产生的机制，这对选择处理方法十分重要。

判断缺失模式使用其中的函数 `aggr()`，它可对缺失数据进行整合，函数结构为：

```
aggr(x, delimiter = NULL, plot = TRUE, ...)
```

`x` 表示一个向量、矩阵或数据框；`delimiter` 用于区分插补变量，如果给出相应的值，其被用来说明变量的值已被插补，但在判断缺失模式时，这一参数默认是忽略的；`plot` 是逻辑值，指明是否绘制图形，默认为 `TRUE`，相当于它内嵌了一个绘图函数。

```
> install.packages("VIM")
> library(VIM)
> aggr(data)
```

结果如图 3.2 所示，图中直接生成两个图像。

第一个图由小条形的长度显示各变量缺失数据比例，由于变量 `PRICE` 和 `SALARY` 都包含缺失，所以图中都对应对应了深色条形；而 15 条记录中有 4 个缺失 `SALARY`，缺失比例超过 25%，因此 `SALARY` 对应的条形长度大于 0.25。

第二个图显示了综合的缺失模式，可以与 `md.pattern()` 生成的结果对照观察，其中浅色方框表示完整数据，深色框表示缺失值。底部的颜色框高度反映了相应组合的频率。

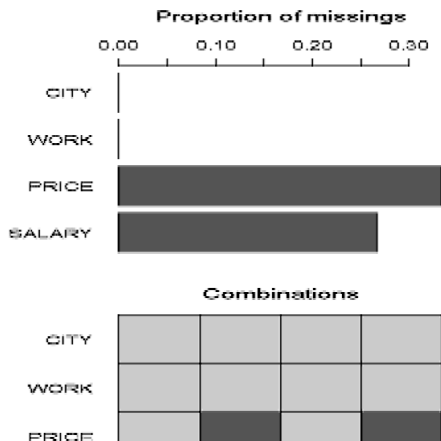


图 3.2 数据缺失模式的绘图

如数据 `data` 中有 8 条完整记录，反映在第一行全部为浅色，底部的蓝色长方形宽为 8；有 2 条记录缺失 `PRICE` 和 `SALARY`，反映在第四行，`PRICE` 和 `SALARY` 对应的方框都是深色，并且底部的深色长方形宽为 2。

3.3.3 处理缺失数据

在有缺失数据的情况下进行的数据分析是不可靠的，并且有些统计方法对数据质量要求很高，不允许数据有遗漏，这就要求我们在数据预处理过程中考虑缺失值的问题。处理缺失数据主要有三种方法。

(1) 删除缺失样本

过滤掉缺失样本是最简单的方式，其前提是缺失数据的比例较少，而且缺失数据是随机出现的，这样删除缺失数据后对分析结果影响不大。

R 可以使用 `complete.cases()` 指令选取完整的记录，有缺失值的行则删去不要。

利用它来选取无缺失数据的行非常方便。从下面的例子可以看出，`data` 数据原来有 15 行，去掉 4 个 `SALARY` 缺失的记录后，剩下 11 行。

```
> data1=data[complete.cases(data$SALARY),]
> dim(data1)
[1] 11 4
```

通过函数 `is.na()` 也可以得到同样结果：

```
> data2=data[!is.na(SALARY),]
> dim(data2)
[1] 11 4
```

对于有多个变量缺失的数据，如果想直接删除所有的缺失值，可以通过 `na.omit()` 函数来完成，这样就删去了 `SALARY` 和 `PRICE` 缺失的所有记录，剩下 8 行完整数据。

```
> data3=na.omit(data)
> dim(data3)
[1] 8 4
```

另外，对于有缺失值的数据，一些函数在计算时可以通过参数设置来忽略缺失值，例如 `mean`、`var`、`sum`、`min`、`max` 等，当 `na.rm=TRUE` 时忽略缺失值。

（2）替换缺失值

缺失值不一定要完全删除，最常见的是通过赋值来解决，用变量均值或中位数来代替缺失值，这样做的优点在于不会减少样本信息，处理起来简单，但缺点在于当缺失数据不是随机出现时会产生偏差。

例如上面缺失工资指数的数据，可以使用已有数据的平均值来替代。通过函数 `is.na()` 来查找缺失值的位置。

```
>data[is.na(data)] = mean(SALARY[!is.na(SALARY)]) #mean 函数是对非 NA 值的 SALARY 数据求平均
```

读入数据时，如使用 `read.table()`，也可以直接通过 `na.strings`=参数来指定代表缺失值的字符串。

（3）多重插补法

多重插补（Multiple Imputation）是用于填补复杂数据缺失值的一种方法，该方法通过变量间关系来预测缺失数据，利用蒙特卡罗随机模拟方法生成多个完整数据集，再对这些数据集分别进行分析，最后对这些分析结果进行汇总处理。

其基本思想是对每一个缺失值进行插补，重复 m 次后得到 m 个插补值，再利用 m 个插补值估计缺失值。比较有代表性的插补算法有 JM 模型和 FSC（Fully Approaches Specification）。

JM 模型对原始数据的要求很高，需要数据服从多元正态分布。R 中可以处理多元正态数据的缺失值的程序包有：`mvnmle` 提供了最大似然估计方法（ML Estimation），`norm` 提供了期望最大化算法（EM algorithm），`monomvn` 可以估计单调多元正态数据的缺失值。

这里重点介绍 FSC 方法，因为它在没有合适的多维分布时也可以使用，FSC 是基于链式方程的插补方法，因此也称为 MICE（Multiple Imputation by Chained Equations）。它与其他多重插补算法的本质区别是，它在进行插补时不必考虑被插补变量和协变量的联合分布，而是利用单个变量的条件分布逐一进行插补。

在 R 语言中通过程序包 `mice` 中的函数 `mice()` 可以实现该方法，它随机模拟多个完整数据集

并存入 `imp`，再对 `imp` 进行线性回归，最后用 `pool` 函数对回归结果进行汇总。

汇总结果和普通回归相似，`nmis` 表示了变量中的缺失数据个数，`fmi` 表示由缺失数据贡献的变异。

为简化计算，假设数据集中只有变量 `SALARY` 含缺失值，重新读入并替换工资指数大于 65 的数值为 `NA`：

```
> data=read.table("d:/data/salary.txt",header=T)
> names(data)=c("CITY","WORK","PRICE","SALARY")
> attach(data)
> data$SALARY=replace(SALARY,SALARY>65,NA)
```

接下来进行随机模拟，再做多元回归：

```
> imp=mice(data,seed=1) #随机模拟数据
> fit=with(imp,lm(SALARY~WORK+PRICE)) #线性回归
> pooled=pool(fit) #回归结果
> options(digits=3) #显示小数点后三位
> summary(pooled)
```

	est	se	t	df	Pr(> t)	lo 95	hi 95	nmis	fmi	lambda
(Intercept)	96.6177	39.2356	2.46	9.53	0.0346	8.6046	184.6307	NA	0.220	0.0723
WORK	-0.0537	0.0187	-2.87	9.71	0.0173	-0.0956	-0.0118	0	0.207	0.0584
PRICE	0.6048	0.1676	3.61	9.07	0.0056	0.2260	0.9835	0	0.254	0.1053

下面手动将预测的结果计算出来，并替换到 `data` 数据集中。通过多重插补法的回归模拟，数据集就补全了。

```
> data.pre=data[is.na(data$SALARY),][,2:3] #选取缺失样本的 WORK 和 PRICE 值
> data.pre=as.matrix(cbind(rep(1,4),data.pre))
> q=pooled$qbar #通过拟合回归预测 SALARY
> pre=data.pre*%q;pre #预测结果
  [,1]
7  33.8
8  44.7
10 42.7
15 65.2
> index=is.na(data$SALARY)
> data$SALARY[index]=pre #替换缺失值
> data[index,]
```

	CITY	WORK	PRICE	SALARY
7	LosAngeles	2068	79.8	33.8
8	Luxembourg	1768	71.1	44.7

10	NewYork	1942	83.3	42.7
15	Tokyo	1880	115.0	65.2

除了 `mice`，目前许多 R 程序包都提供了多重估算函数，因为不同数据类型要求的估计方法也不相同。

例如程序包 `cat` 可以做分类数据的多重估算，`mix` 适用于分类变量和连续变量的混合型数据，`pan` 提供了面板数据的多重估算等。

3.4 数据整理

数据被清理干净后，可能还需要经过一定的形式变化，才能用于数据分析和绘图。这里主要指数据集的合并与拆分，例如将一个大数据集按照某个特定的条件拆分成几个小的数据集，或将原始数据集中的某些变量挑出或除去，重新放入一个新的数据集中，使得数据集的内容变得简化，便于进行统计分析。

3.4.1 数据合并

在 R 语言中，数据合并有多种形式，这里介绍主要的三类函数。

(1) 函数 `cbind()`、`rbind()`

在 R 中数据的合并由 `cbind()` 和 `rbind()` 这两个基本指令实现，可以在向量、矩阵（`matrix`）、数据框（`data.frame`）和列表上使用。`cbind()` 按列的方式将对象连接到一起，`rbind()` 按照行将对象连接在一起。

在下面的例子中，尤其要注意的一点是，列表可以存储字符型变量，合并时向量 `a` 中既含有字符又含有数字，读入外部文件时应该将 `stringsAsFactors` 设置为 `F`，意思是读入数据时不要自动将字符串转换为因子，若忽略这一设置，合并时将会报错。

```
> a=c("Hongkong",1910,75.0,41.8)
> data1=rbind(data,a)
```

原来的数据集 `data` 有 15 条记录，使用 `rbind` 指令后，就增加了第 16 行。

```
>data1[14:16,]
      City Work Price Salary
14  Taipei 2145  84.3   34.5
15   Tokyo 1880  115    68
16 Hongkong 1910   75   41.8
```

(2) 构造 `data.frame`

对数据“整容”最简单的思路是把数据向量化，再按要求用向量构建其他类型的对象。一些结构相似的对象，如向量（数值型、字符型、逻辑型）、因子、数值矩阵、列表或其他数据框等，

可以被合并为一个数据框。

例如，将 3 个向量合并为一个数据框：

```
> weight=c(150,135,210,140) #数值型向量
> height=c(65,61,70,65)
> gender=c("F","F","M","F") #字符型向量
> stu=data.frame(weight,height,gender)
```

合并时，变量名称就自动变成了新数据框的列名，也可以用 `names()` 重新给其赋值。`row.names()` 用于对数据框的行进行命名。

```
> row.names(stu)=c("Alice","Bob","Cal","David")
```

从数据框内取行或列有多种方式，既可以通过行号、列号来获取，也可以通过行名、列名来获取，尤其是列的获取方式非常多样，下面列出了 4 种：

```
> stu[, "weight"]
[1] 150 135 210 140
> stu$Weight # "$"用于取列
[1] 150 135 210 140
> stu[["weight"]] #双括号+名称
[1] 150 135 210 140
> stu[[1]] #双括号+下标，用于数据框和列表数据的获取
[1] 150 135 210 140
> stu["Cal",] #获取行
weight height gender
Cal      210      70      M
> stu[1:2,1:2]
weight height
Alice    150     65
Bob      135     61
```

(3) 函数 merge()

在 R 中合并两个数据集可以通过专门的函数 `merge()` 来实现。`merge` 通过相同的列或行名来识别，合并两个数据框或列表，其调用格式如下，参数如表 3.3 所示。

```
merge(x, y, by = intersect(names(x), names(y)), by.x = by, by.y = by, all = FALSE,
      all.x = all, all.y = all, sort = TRUE, suffixes = c(".x", ".y"), incomparables = NULL, ...)
```

表 3.3 merge()的参数设置

参 数	含 义
x, y	要合并的数据集
by	指定合并的依据（相同的行或列）
by.x, by.y	分别为第一个数据框和第二个数据框要连接的列名

续表

参 数	含 义
all, all.x, all.y	逻辑值，默认为 FALSE。以 all.x=TRUE 为例，表示当 x 中的行没有相应的 y 进行匹配时，用 NA 填充；若为 FALSE，那么仅输出 x 和 y 中都包含的行

假设建立另一个数据集 `index`，其包含各个城市名称及相应的序号，我们将要把 `index` 合并到之前的 `data` 数据集中，以相同的变量 `City` 为合并的依据，合并后将在 `data` 数据集的基础上增加新的一列 `Index`。

```
>index=list("City"=data$City,"Index"=1:15)
>index
$City
 [1] Amsterdam Bombay    Chicago    Dublin     Frankfurt London    LosAngeles
Luxembourg MexicoCity
[10] NewYork    Paris      Singapore Sydney     Taipei    Tokyo

$Index
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
>data.index=merge(data,index,by="City")
>data.index
      City      Work      Price      Salary      Index
1  Amsterdam    1714     65.6     49.0         1
2    Bombay    2052     30.3      5.3         2
3    Chicago    1924     73.9     61.9         3
4    Dublin    1759     76.0     41.4         4
5  Frankfurt    1650     74.5     60.4         5
...
```

3.4.2 选取数据的子集

有时数据要合并，有时数据却要拆分，尤其当数据量很大的时候，可能只需要提出其中的一部分做分析。在 R 中，选取数据子集用中括号[]。

下面以 `data` 数据集为例来说明。如果想选取工资指数大于 65 的城市数据，就可以用如下指令实现。

需要注意的是，中括号内逗号的前面表示行指标，即选取适合的行；后面是列指标，为空表示所有的列都选取。

```
>data[data$Salary>65,]
      City      Work      Price      Salary
7  LosAngeles    2068     79.8     65.2
8  Luxembourg    1768     71.1     71.1
10   NewYork    1942     83.3     65.8
15    Tokyo    1880    115.0     68.0
```


选取第二行和第四行：

```
>data[c(2,4),]
      City Work Price Salary
2  Bombay 2052  30.3    5.3
4  Dublin 1759  76.0   41.4
```

选取价格指数等于 65.6 的行，注意要用双等号==。

R 中用于表示数量关系的符号还包括：不等于!=、小于<、大于等于>=和小于等于<=。

```
>data[data$Price==65.6,]
      City Work Price Salary
1  Amsterdam 1714  65.6    49
```

3.4.3 数据排序

R 中的排序函数 `sort()` 只能对向量进行简单的排序，对含有多变量的数据集，如对 `data` 按照工资指数排序时，需要用 `order` 指令来完成，其调用格式如下：

```
order(..., na.last = TRUE, decreasing = FALSE)
```

其中，`na.last` 控制对缺失值 NA 的处理，默认为 FALSE，即排序时将缺失值放在最前；如果为 TRUE，则排序时将缺失值放在最后；如果 `na.last=NA`，将删除缺失值。`decreasing` 用来设置升序或降序，默认为 FALSE，也就是由小至大排列。

```
>order.salary=order(data$Salary)
>order.salary
[1] 2 9 12 14 4 11 6 1 13 5 3 7 10 15 8
```

函数 `sort.list` 与 `order` 的作用相同：

```
>sort.list(data$Salary)
[1] 2 9 12 14 4 11 6 1 13 5 3 7 10 15 8
```

输出的结果说明第 2 行的工资指数最小，紧接着是第 9 行、第 12 行……。使用中括号选取数据，就能完成排序。

```
>data[order.salary,]
      City Work Price Salary
2  Bombay 2052  30.3    5.3
9  MexicoCity 1944  49.8    5.7
12 Singapore 2042  64.4   16.1
14  Taipei 2145  84.3   34.5
4  Dublin 1759  76.0   41.4
11  Paris 1744  81.6   45.9
6  London 1737  84.2   46.2
1  Amsterdam 1714  65.6   49.0
13  Sydney 1668  70.8   52.1
5  Frankfurt 1650  74.5   60.4
```

3	Chicago	1924	73.9	61.9
7	LosAngeles	2068	79.8	65.2
10	NewYork	1942	83.3	65.8
15	Tokyo	1880	115.0	68.0
8	Luxembourg	1768	71.1	71.1

指令 `order` 返回向量排序后各数字的原始位置，与之非常相关的指令是秩 (`rank`)，它返回每个数字在整个向量中的秩，可以简单地理解为各个数字的大小顺序。例如：

```
>rank(data$Salary)
[1] 8 1 11 5 10 7 12 15 2 13 6 3 9 4 14
```

说明第一行的工资指数排在第 8，第二行的工资指数最小。

3.5 长宽格式的转换

有时为了计算方便，需要将原始数据的行列进行调换，这就是转置功能。

这在 R 中实现起来也非常简单，只需要一个指令 `t()` 就可以了，原来的列变量 CITY、WORK 等在转置后就变成了行向量。

```
>t(data)
      [,1]      [,2]      [,3]      [,4]      [,5]      .....
CITY  "Amsterdam" "Bombay" "Chicago" "Dublin" "Frankfurt".....
WORK  "1714"      "2052"   "1924"   "1759"   "1650"   .....
PRICE " 65.6"      " 30.3"   " 73.9"   " 76.0"   " 74.5"   .....
SALARY "49.0"      " 5.3"   "61.9"   "41.4"   "60.4".....
```

做转置很简单，这里也只是抛砖引玉，接下来要介绍的几个函数在数据汇总中非常好用，通过几个指令就可以把长格式的数据变换为短格式的数据框。

3.5.1 揉数据函数

R 中有两个揉数据函数 `stack()` 和 `unstack()`，用于数据长格式和宽格式之间的转换，我们形象地比喻其为“揉数据”，也就是在原有的数据格式基础上，将其变换成其他的形式。

`stack()` 把一个数据框转换成两列：一列为数据，另一列为数据对应的列名称。

```
> x=data.frame(A=1:4,B=seq(1.2,1.5,0.1),C=rep(1,4))
>x
  A  B C
1 1 1.2 1
2 2 1.3 1
3 3 1.4 1
4 4 1.5 1
>x1=stack(x)
```

```
>x1
valuesind
1 1.0 A
2 2.0 A
3 3.0 A
4 4.0 A
5 1.2 B
6 1.3 B
7 1.4 B
8 1.5 B
9 1.0 C
10 1.0 C
11 1.0 C
12 1.0 C
```

`unstack()`是 `stack` 的逆过程，被转换的对象包含两列，它把数据列按照因子列的不同水平重新排列，分离为不同的列。

很早以前我们觉得 Excel 的数据透视表是一个非常神奇的工具，其实在 R 里面 `unstack()`就可以实现类似的效果。如果各列的数量相等，`unstack()`就将它强制转换为数据框，否则转换为列表。

被转换的对象默认第一列为数据，第二列为因子，如果不是这个顺序，则需要参数 `form` 设置公式类型：公式左边的变量是值，右边的变量会被当成因子类型，它的每个水平行都会形成一列。

```
>unstack(x1,form=values~ind)
  A  B  C
1 1 1.2 1
2 2 1.3 1
3 3 1.4 1
4 4 1.5 1
```

和 `stack` 作用类似的是函数 `reshape()`，但该函数里的参数非常多，很难记忆，使用起来也不方便，尤其当“大牛”Hadley Wickham 创造出程序包 `reshape` 和 `reshape2` 后该函数几乎不再被提起。既然可以被遗忘，我们就等走投无路的时候（虽然几乎不会发生）再考虑用它吧，下面直接进入新的程序包。

3.5.2 揉数据的最佳伴侣

揉数据的最佳伴侣是 `reshape2` 程序包，提到这个程序包的作者 Hadley Wickham 也许你并不熟悉，但了解了他写过的 R 程序包之后，你一定会承认他是个“大牛”。

Wickham 写过很多程序包，他的想法新颖，包中的函数既方便使用又能产生很好的效果，可以说颠覆了许多传统的函数方法，不可不谓牛人也。最著名的有 `ggplot2`、`plyr`、`reshape2` 等，在下一章中我们会重点介绍 `ggplot2`——目前备受瞩目的画图工具，`plyr` 在数据处理方面也非常好用。

程序包 `reshape2` 是 `reshape` 的重写版，是专门用于数据集形状转换的，可以直接安装和使用 `reshape2`。包里的函数很少，一般用户常使用 `melt()`、`acast()` 和 `dcast()`，但它们却可以把数据“揉”成各种形状。

`melt` 本身的意思是溶解、分解，其作用在一个数据集上其实就是拆分数据，它的对象可以是数组（array）、数据框或列表。与 `stack()` 的作用一样，`melt` 将数据框或列表转换成两列，只是这里第一列是变量名称，第二列是变量数值。

```
>install.packages("reshape2")
>library("reshape2", lib.loc="D:/R-3.0.1/library")
>melt(x)
Using as id variables
variable value
1      A  1.0
2      A  2.0
3      A  3.0
4      A  4.0
5      B  1.2
...
```

`melt()` 还可以用于处理更复杂的数据框，不过需要进行参数设置：

```
melt(data, id.vars, measure.vars, variable.name = "variable", ..., na.rm = FALSE,
      value.name = "value")
```

其中，`id.vars` 是被当做新数据（长格式）维度的列变量，可以指定多个，成为 `variable` 列；`measure.vars` 是观测值的列变量，构成新数据的 `value` 列；分别用 `variable.name` 和 `value.name` 来指定新的列变量名。当然这两个参数可以只指定其中一个，剩余的列自动归为另一个参数。

下面使用基础包中的 `airquality` 数据集来举例说明，以变量月、日为分组依据，将原始数据转换为长格式的数据集。

```
>data(airquality)
>str(airquality) #显示R对象的内部结构，功能类似于summary()
'data.frame':  153 obs. of  6 variables:
 $ Ozone  :int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  :int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
>longdata=melt(airquality,id.vars=c("Ozone","Month","Day"),measure.vars=2:4)
```

```
>str(longdata)
'data.frame':   459 obs. of  5 variables:
 $ Ozone   : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Month   : int   5 5 5 5 5 5 5 5 5 5 ...
 $ Day     : int   1 2 3 4 5 6 7 8 9 10 ...
 $ variable: Factor w/ 3 levels "Solar.R","Wind",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ value   : num  190 118 149 313 NA NA 299 99 19 194 ...
```

看到这里，很多人可能会产生疑问：用 `melt()` 处理后的数据变长了，又有什么好处呢？对数据格式的重新排列可以根据实际中的需要进行，在这个例子中，拆分以后的数据可以按多个不同变量分类，利用 `ggplot2` 在一个图形中多维度地展示 `value` 值。

```
>library(ggplot2)
> p=ggplot(data=longdata,aes(x=Ozone,y=value,color=factor(Month)))
>p+geom_point(shape=20,size=4)+facet_wrap(~variable,scales="free_y")+
  geom_smooth(aes(group=1), fill="gray80") #scale="free_y"设置每个图形自动调整 y 轴范围
```

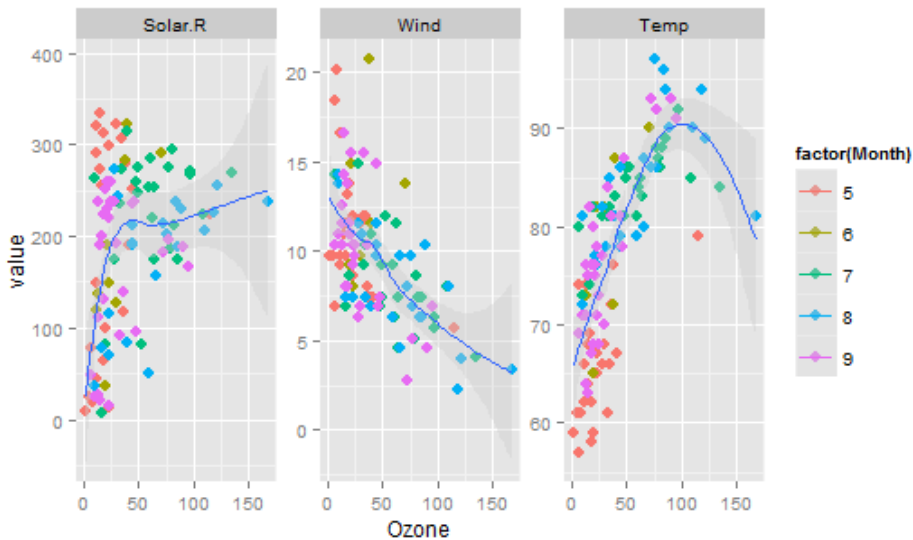


图 3.3 ggplot2 绘图

和 `stack()` 一样，`melt()` 也有对应的函数用来还原数据：`acast()` 用于数组，`dcast()` 用于数据框，其中的参数 `formula` 是一个公式，左边的每个变量都会成为新数据集中的一列，右边的变量是因子，其每个水平行在新数据集中成为一列，从而把长格式数据转换为短格式。

包 `reshape2` 之所以厉害，在于它不仅仅可以转换数据，还可以对数据进行汇总，由参数 `fun.aggregate` 实现汇总，例如：

```
>shortdata=dcast(longdata,formula=Ozone+Month+Day~variable)
>head(shortdata,5)
```

	Ozone	Month	Day	Solar.R	Wind	Temp
1	1	5	21	8	9.7	59
2	4	5	23	25	9.7	61
3	6	5	18	78	18.4	57
4	7	5	11	NA	6.9	74
5	7	7	15	48	14.3	80

基本分析及应用

- 第 4 章 数据的图形描述
- 第 5 章 数据的描述性分析
- 第 6 章 参数估计及 R 实现
- 第 7 章 假设检验及 R 实现
- 第 8 章 方差分析及 R 实现
- 第 9 章 回归分析及 R 实现
- 第 10 章 主成分分析与因子分析
- 第 11 章 典型相关分析和对应分析
- 第 12 章 判别分析和聚类分析
- 第 13 章 时间序列分析及 R 实现

第 4 章

数据的图形描述

图形是展示数据分析结果的一个重要工具，它可以直观并且美观地反映变量之间的各种关系。作为数据分析的重要平台，R 软件拥有非常强大的绘图功能，而且工具的灵活性很强，用户可以根据自己的要求绘制所需图形。

随着越来越多的绘图程序包的开发，用 R 画出来的图形也越来越美观。学会用 R 画图，是用 R 做数据分析必不可少的一个前提。

4.1 R 绘图概述

R 提供了非常丰富的绘图功能，其绘图函数多种多样，用户可以通过 R 自身提供的演示示例来初步了解。通过以下两个函数，可以分别展示二维、三维图形的示例。

```
>demo(graphics)
>demo(persp)
```

在 R 中执行绘图命令，会启动一个图形设备驱动，该驱动会打开特定的图形窗口以显示交互式的图片。一旦设备驱动启动，R 绘图命令就可以被用来产生统计图或者设计全新的图形显示。此外，R 有一系列图形参数，可以修改这些图形参数从而定制你的图形环境。

图形工具是 R 环境的一个重要组成部分。R 提供了多种绘图相关的命令，可分成三类：

- 高级绘图命令。在图形设备上产生一个新的图区，它可能包括坐标轴、标签、标题等。
- 低级绘图命令。在一个已经存在的图形上加上更多的图形元素，如额外的点、线和标签。
- 交互式图形命令。允许交互式地用鼠标在一个已经存在的图形上添加图形信息或者提取图形信息。

使用 R 语言作图，主要按照以下步骤进行：

- ① 获取原始数据，准备好绘图需要的变量。
- ② 如有需要，对绘图区域进行设置、分割。
- ③ 绘制图形，例如创建坐标轴并绘制点图、曲线或其他类型的图。
- ④ 标注图形。对图形进行标注，包括在图形中添加标题、坐标轴标注、文字标注等。
- ⑤ 设置图形格式，添加图例。包括设置图形中的线宽、线型、颜色，标记点的形状、大小、颜色，以及坐标轴格式等。
- ⑥ 保存和导出图形。按指定文件格式、属性保存或导出图形，以备以后使用。

接下来我们按照这个流程，一步一步地了解 R 绘图的各种功能。

4.2 绘图区域分割

数据的获取和准备在前面两章已经讲过，通常 R 作图的对象是向量或数据框，读入数据时一般可以直接形成图形。下一步是对绘图区域的设置，默认情况下，绘图区域每次只显示一个图，如果想要做图形的比较，那最好可以在一个界面内同时显示多个图框，这就需要进行设置。在 R 中，主要有三个函数 `par()`、`layout()` 和 `split.screen()` 可以完成图形区域的分割，下面分别进行讲述。

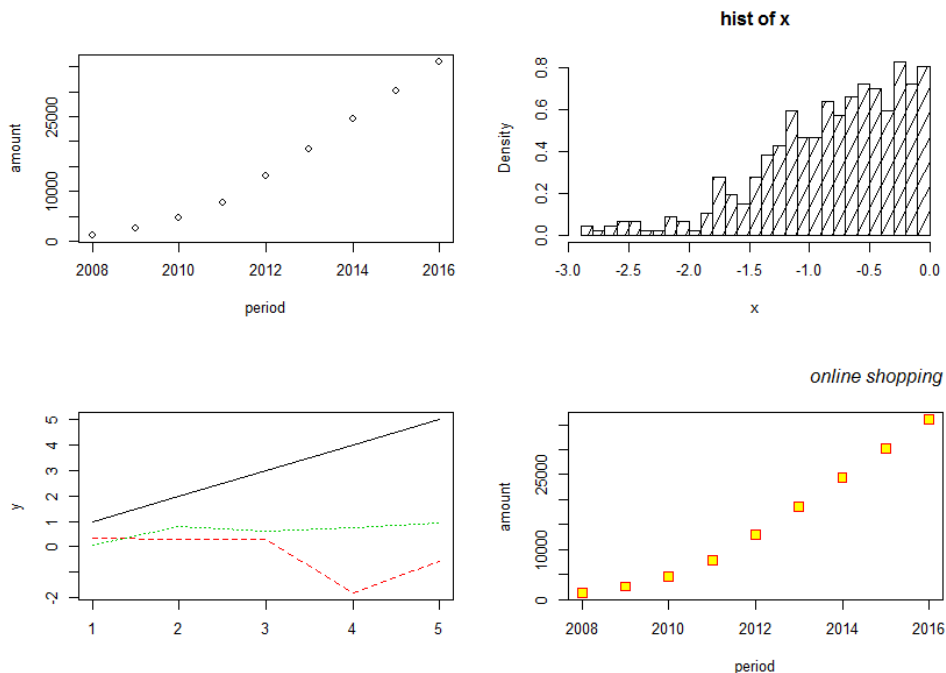
4.2.1 函数 `par()`

函数 `par()` 可以将绘图区域分割成规则的几部分，例如 `par(mfrow=c(3,2))` 将图形区域分成 3×2 的多重图框，每块显示一个图形，按行显示，也可以使用 `mfcoll` 按列输入图形。

例如，我们想把 4 幅图画在同一个图形区域中，可以使用如下指令完成，涉及的绘图函数下文会详细介绍，这里主要说明如何分割图形区域。

```
>#首先，准备绘图数据：从外部读取或随机数生成
>dat=read.table("d:/data/online shopping.txt",header=T)
>attach(dat)
> x=rnorm(1000)
> x=x[x<0]
> y=data.frame(x1=1:5,x2=rnorm(5,0,1),x3=rgamma(5,2,3))
#将区域分为4个部分
>par(mfrow=c(2,2))
#分别输入4个绘图命令
>plot(period,amount)
>hist(x,xlim=range(x),main="hist of x",freq=F,nclass=30,density=20,angle=45)
>matplot(y,type="l",col=1:3)
>plot(period,amount,pch=22,col="red",bg="yellow",cex=1.5)
>title("online shopping",font.main=3,adj=1)
```

绘制的图形如图 4.1 所示。

图 4.1 使用函数 `par()` 分割绘图区域

使用指令 `par` 分割的区域比较规则，想要灵活度就要与指令 `mfg` 配合，但频繁使用 `mfg` 并不方便，因此我们介绍函数 `layout()`。

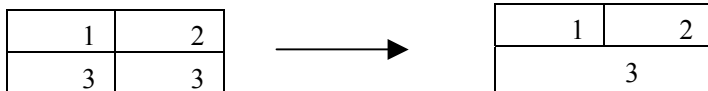
4.2.2 函数 `layout()`

`layout()` 内部的参数是一个矩阵（`matrix`），其通过定义矩阵来灵活地将图形区域进行分割，`matrix` 默认按列输入。

```
layout(mat, widths = rep.int(1, ncol(mat)), heights = rep.int(1, nrow(mat)), respect = FALSE)
```

`mat` 为矩阵，用于设置窗口的划分，矩阵的 0 元素表示该位置不画图，非 0 元素必须包括从 1 开始的连续整数值，比如， $1, 2, \dots, N$ ，按非 0 元素的大小设置图形的顺序。`widths` 用来设置窗口不同列的宽度，`heights` 设置不同行的高度。例如：

- `layout(matrix(1:4,2,2))`，将绘图区域分成 2×2 的多重图框。
- `layout(matrix(c(1,3,2,3),2,2))`，将图形区域分成三个不规则的区域。



- `layout(matrix(c(1,1,2,3,2,3),2,3))`，将图形区域分成如下的不规则区域。

1	2
	3

分割完成后，通过指令 `layout.show(3)` 可以查看区域分割后的结构。要取消图形区域分割，输入指令 `layout(1)`。

4.2.3 函数 `split.screen()`

`split.screen()` 同样由向量或矩阵灵活控制区域的分割方式，例如：

```
>split.screen(c(2,1)) #将图形区域分成上下两部分显示
[1] 1 2
>split.screen(c(1,2),screen=2) #将第二部分（下半区）又分割成两个区域
[1] 3 4
```

从而，现在将图形区域分成了三块，原来的区域 2 分割后编号分别为 3 和 4。

```
> screen(1) #准备在第一个区域绘图
>hist(x,xlim=range(x),main="hist of x",freq=F,nclass=30,density=20,angle=45)
> screen(3) #准备在下半区的第一个区域绘图
>matplot(y,type="l",col=1:3)
> screen(4) #准备在下半区的第二个区域绘图
>plot(period,amount,pch=22,col="red",bg="yellow",cex=1.5)
```

分区结果以及所绘制图形如图 4.2 所示。

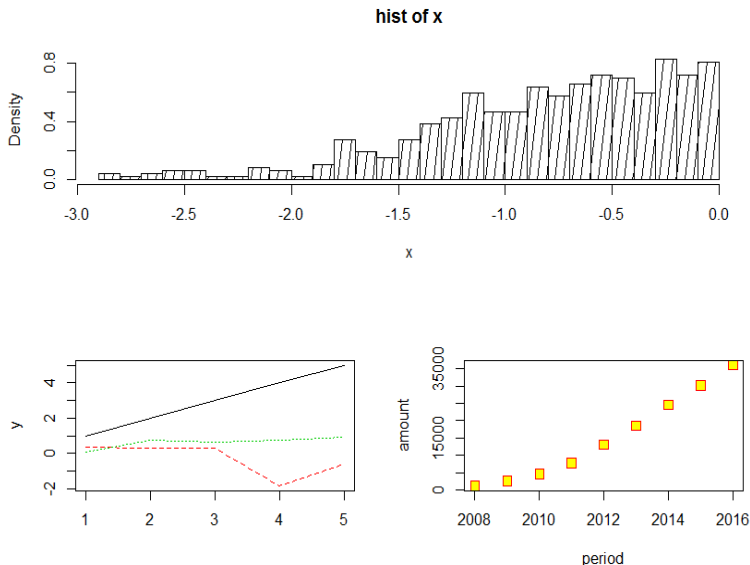


图 4.2 使用函数 `split.screen()` 分割绘图区域

在每个区域画图后，还可以通过指令 `erase.screen()` 清除图形，然后重新绘制。使用指令 `close.screen(all=TRUE)` 可退出分割画面的模式。

4.3 二维图形

设置好绘图区域，下一步就可以通过 R 中的各种绘图函数将数据可视化了。众所周知，图形是我们直观地了解和认识数据的一种可视化方法，如果能将观察数据直接显示在一个平面图中，我们就可以一目了然地看出分析变量之间的数量关系。

R 具有多种作图工具，我们首先从基本的二维图形开始介绍，例如散点图、直方图等就是比较常用的二维平面图示。

4.3.1 高级绘图函数

在 R 中有两种绘图函数，分别为：

- 高级绘图函数。创建一个新的图形。
- 低级绘图函数。在已经存在的图上添加更多元素，如额外的点、线和图例等。

绘制图形的第一步，要从高级函数开始，创建坐标轴并绘制点图、曲线或其他类型的图。

1. 函数 `plot()`

`plot()` 是最常用的高级绘图函数，这是一个泛型函数，其产生的图形依赖于参数的类型。`plot` 可以有多种变形，具体如表 4.1 所示。

表 4.1 `plot()` 函数

函 数	功 能
<code>plot(x)</code>	以 <code>x</code> 的值为纵坐标、序号为横坐标绘图
<code>plot(x,y)</code>	<code>x</code> 值为横坐标、 <code>y</code> 值为纵坐标的散点图（如果 <code>x</code> 是一个时间序列，则产生时间序列图）
<code>plot(f)</code>	<code>f</code> 是一个因子（factor）对象，产生 <code>f</code> 的直方图
<code>plot(f,y)</code>	<code>f</code> 是一个因子对象， <code>y</code> 是数值向量，产生 <code>y</code> 在 <code>f</code> 的各水平下的箱线图
<code>plot(data.frame)</code>	产生数据框中各变量的分布图
<code>plot(~a+b+c)</code>	对于形如 <code>a+b+c</code> 的表列，产生各对象的分布图
<code>plot(y~a+b+c)</code>	<code>y</code> 可以是任何对象，分别产生 <code>y</code> 相对于 <code>a,b,c</code> 各对象的图

下面以 2008~2016 年各年我国网络购物交易规模数据为例，其中 2013~2016 年数值为预测结果。使用 `plot` 函数绘制简单的散点图。

```
>dat=read.table("d:/data/online shopping.txt",header=T)
>attach(dat)
>plot(period,amount)
```

绘制结果如图 4.3 所示。

在 plot 函数中指明所使用的数据集，还可通过如下方式，实现与上述指令相同的效果：

- plot(dat\$period,dat\$amount)
- with(dat,plot(period,amount))
- plot(amount~period,data=dat)

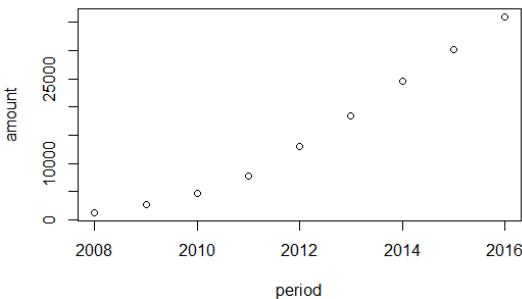


图 4.3 网络购物交易规模散点图

除了 plot()函数，还有许多高级绘图函数用来绘制各种类型的图形，常用的如表 4.2 所示。

表 4.2 其他高级绘图函数

函 数	功 能
hist()	直方图
pie()	饼图
boxplot()	箱线图
barplot()	条形图
qqnorm()	正态分位图
heatmap()	热图
stripchart()	将对象的值画在一条线段上，样本量较小时可以代替箱线图
stars()	对象为矩阵或数据框，绘制星图
symbols(x,y)	以 x 和 y 指定坐标位置画出符号（圆、正方形、星形或箱线图），符号的大小颜色由另外的变量确定

本章开头提到，绘图的第④步是标注图形，即在图形中添加标题、坐标轴标注、文字标注等，这是通过设置绘图参数来完成的。高级绘图函数有一些共同的参数设置，具体如表 4.3 所示。

表 4.3 高级绘图函数的参数设置

参 数	含 义
add	默认为 FALSE。若为 TRUE，则将图形叠加到前一个图上（若之前已绘制了一个图）
axes	默认为 FALSE，不绘制轴和边框

续表

参 数	含 义
type	设置图形的类型：p 为点、l 为线、b 为点连线式、o 为点线式且线在点上、h 为垂直线、s 为阶梯式，垂直线顶端显示数据。S 为阶梯图，线底端显示数据，n 为不显示图形，只画出坐标轴，常与低级绘图函数一起使用
main	主标题
sub	副标题
xlab, ylab	坐标轴的标签
xlim, ylim	指定坐标轴的显示范围，例如 xlim=c(1,10)或者 xlim=range(x)

互联网购物和电商行业是目前的热点问题，我们使用 2012 年各购物网站交易规模的市场份额数据绘制饼图，由图 4.4 可以直观地看出各网站份额的占比情况，并为其添加主标题。

```
> percent=c(56.7,19.6,5.5,4.7,2.7,10.8)
> labels=c("天猫","京东","苏宁易购","腾讯 B2C","亚马逊中国","其他") #添加标签
> pie(percent,labels,col=2:7,main="Pie chart for online shopping",font=2)
```

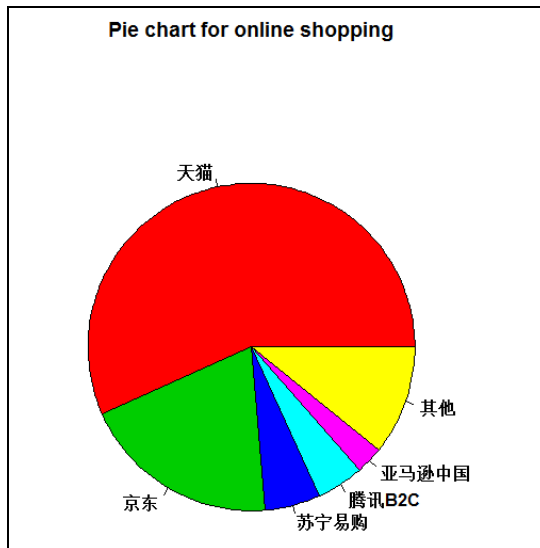


图 4.4 表示各网站市场份额的饼图

函数 hist()

在统计数据的描述分析中，我们常常使用直方图来观察数据的分布状态，因此在这里重点介绍直方图的绘制。R 软件中使用 hist() 指令绘制直方图。

```
hist(x, breaks = "Sturges", freq = NULL, probability = !freq,
include.lowest = TRUE, right = TRUE, density = NULL, angle = 45,
col = NULL, border = NULL, main = paste("Histogram of" , xname),
```

```
xlim = range(breaks), ylim = NULL, xlab = xname, ylab, axes = TRUE,
plot = TRUE, labels = FALSE, nclass = NULL, warn.unused = TRUE, ...)
```

除了基本的参数，hist()有一些特殊的参数设置，具体如表 4.4 所示。

表 4.4 hist()的参数设置

参 数	含 义
breaks	设置每个柱形的间距
freq	若为 FALSE 表示 y 轴（柱形高度）为密度而不是频率
probability	若为 TRUE 表示 y 轴是频率而不是频数
nclass	改变分类数
density 和 angle	设置柱形上的斜线
border	设置柱形边界的颜色

下面举例说明如何使用 hist()绘制直方图。首先在 R 中随机生成 1000 个正态分布的随机数，然后使用指令 hist 画图。

```
> x=rnorm(1000)
> x=x[x<0]
> hist(x,xlim=range(x),main="hist of x",freq=F,nclass=30,density=20,angle=45)
```

绘制的结果如图 4.5 所示。

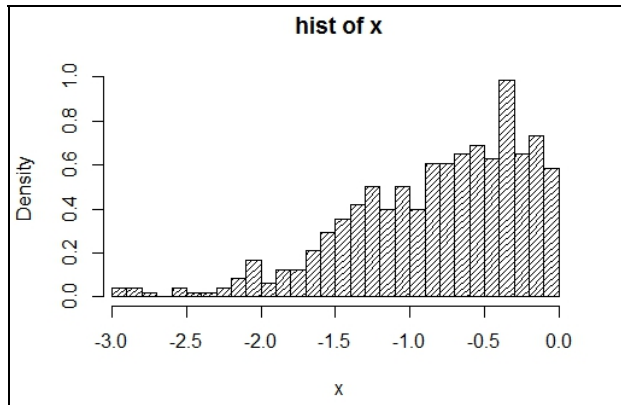


图 4.5 直方图

4.3.2 多元数据绘图

在实际的数据分析中，几乎我们得到的所有数据集都是多变量的，这对图形描述提出了更高的要求。接下来介绍几个 R 中常用的多元作图工具（见表 4.5），它们也属于高级绘图函数的范畴。

表 4.5 多元绘图函数

函 数	功 能
matplot(x)	x 为数据框，将其各列都绘制在同一张图中
matplot(x,y)	二元作图：x 的各列与 y 的各列对应
coplot(x~y c)	关于 c 的每个数值绘制 x 与 y 的散点图
coplot(x~y c+d)	在 c 和 d 的联合区间内绘制 x 与 y 的散点图
Pairs(x)	x 是矩阵或数据框，产生 x 各列之间的散点图，共 $n(n-1)$ 个图

其中，函数 `coplot()` 和 `pairs()` 默认的图形样式为散点图，但通过参数 `panel`，可以设置为期望的图形。`maplot()` 通过参数 `type` 可修改图形样式。接下来，我们通过两个简单的例子来说明多元数据的绘图，首先使用 `matplot()` 对数据框作图，可以将数据框内的各变量画在同一个图形区域中。

```
> # 首先生成数据；x2 为正态分布随机数，x3 为 gamma 分布的随机数
> y = data.frame(x1 = 1:5, x2 = rnorm(5, 0, 1), x3 = rgamma(5, 2, 3))
> matplot(y, type = "l", col = 1:3, lwd = 2)
# 添加图例
> legend(1, 5, col = 1:3, pch = "——", legend = c("x1", "x2", "x3"))
```

绘制结果如图 4.6 所示。

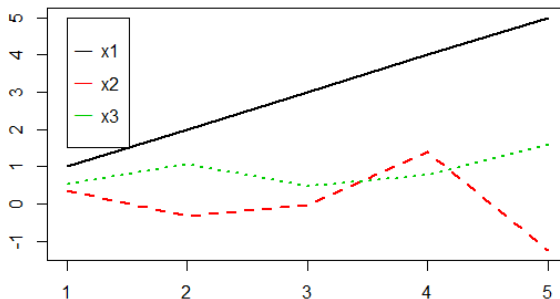


图 4.6 matplot()多元绘图

当多元数据中含有因子（factor）变量时，使用 `coplot()` 函数作图更为合适。以 R 中自带的数据集 `warpbreaks` 为例，其中包含三个变量：`breaks` 为数值型向量，`wool` 和 `tension` 为因子。

```
> data(warpbreaks)
> coplot(breaks ~ 1:54 | wool * tension, data = warpbreaks, col = "red", bg = "pink",
pch = 21, bar.bg = c(fac = "light blue"))
```

绘制结果如图 4.7 所示。

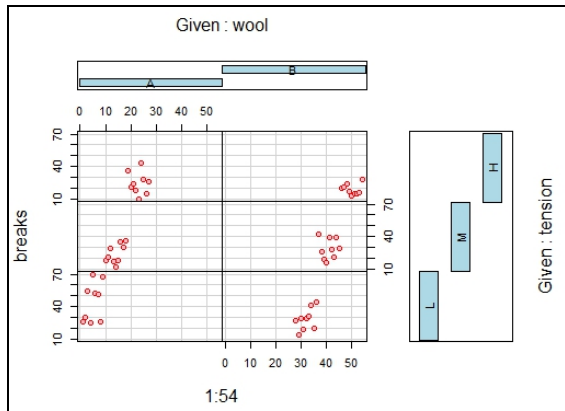


图 4.7 coplot()多元绘图

4.3.3 低级绘图函数

利用高级函数画出基本图形后，可使用低级绘图函数添加新的图形元素，如点、图例、标签等。表 4.6 列出了主要的函数。

表 4.6 低级绘图函数

函 数	功 能
points(x,y)	添加点
lines(x,y)	添加线
title()	添加标题
legend(x,y,legend)	在点(x,y)处添加图例 legend
text(x,y,labels)	在坐标(x,y)处添加文字，用 labels 指定文字内容。常用 text 在图中添加数学表达式，需要通过 expression 转换数学公式
mtext(text,side,line)	在图的边缘添加文本 text，side 指定添加在哪一边，line 指定文字离图形区域的行数
axis(side,vect)	画坐标轴
segments(x0,y0,x1,y1)	从点(x0,y0)到点(x1,y1)添加线段
arrows(x0,y0,x1,y1)	从点(x0,y0)到点(x1,y1)添加箭头
abline(a,b)	绘制斜率为 b、截距为 a 的直线；若参数为 h=y，则表示在纵坐标 y 处画水平线；若参数为 v=x，则表示在横坐标 x 处画垂直线
rect(x1,y1,x2,y2)	以(x1,y1)为左下角、(x2,y2)为右上角绘制长方形
polygon(x,y)	绘制多边形
box()	在图形上添加边框
rug(x)	在 x 轴上用短线标出数据横坐标的位置

下面使用低级函数，我们在上面直方图例子的基础上，添加线和图例：

```
>lines(density(x),col="blue")
```

```
>lines(-3:0,dnorm(-3:0,mean(x),sd(x)),col="red")
>legend(-3,1,pch=c(15,-1,-1),lty=c(-1,1,1),col=c("gray","blue","red"),legend=c("h
istogram","density line","normal density line"))
```

绘制结果如图 4.8 所示。

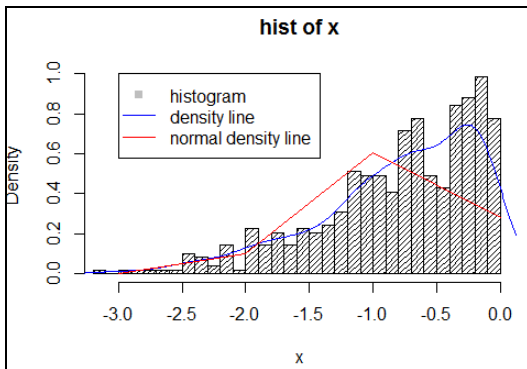


图 4.8 添加线和图例

4.3.4 图形美化

使用高级绘图函数可以描绘出基本图形，函数内还包括大量的参数用于指定图形的颜色、形状等，通过改变这些设置可以对图形进行美化。通过指令 `help(par)` 可以查看这些参数的详细信息，表 4-7 中列出了常用的一些参数。

表 4.7 绘图函数中的参数设置

参 数	含 义
adj	控制文本对齐方式：0 是左对齐，0.5 是居中对齐，1 是右对齐
bg	背景颜色， <code>colors()</code> 可以显示 657 种可用颜色的名称
bty	图形边框形状
cex	符号和文字大小。 <code>cex.axis</code> 、 <code>cex.lab</code> 、 <code>cex.main</code> 、 <code>cex.sub</code> 分别控制坐标轴、标签、标题和副标题的文字大小
col	颜色。与 <code>cex</code> 类似，也有 <code>col.axis</code> 、 <code>col.lab</code> 、 <code>col.main</code> 和 <code>col.sub</code> 参数
font	文字字体：1 为正常，2 为斜体，3 为粗体，4 为粗斜体
las	坐标轴刻度数字的方向：0 为平行于轴，1 为横排，2 为垂直于轴，3 为竖排
lty	线型：1 为实线，2 为虚线，3 为点线，4 为点虚线，5 为长虚线，6 为双虚线。也可以自定义
lwd	线的宽度
pch	指定绘图符号，见图 4.9。也可以用 " " 自定义符号
ps	文字大小
xaxt,yaxt	若 <code>xaxt="n"</code> ，说明不显示 x 轴

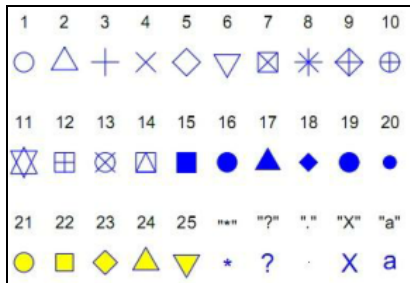


图 4.9 绘图符号 (pch)

使用函数 `par()` 可以永久改变绘图参数，即后面的图形都按照 `par` 指定的样式绘图。例如输入 `par(col="red")` 将后面图形的符号颜色都指定为红色。

最后根据上述的参数设置，我们对第一个例子——网络购物交易规模的散点图进行美化。

```
>plot(period,amount,pch=22,col="red",bg="yellow",cex=1.5)
>title("online shopping",font.main=3,adj=1)
```

结果如图 4.10 所示。

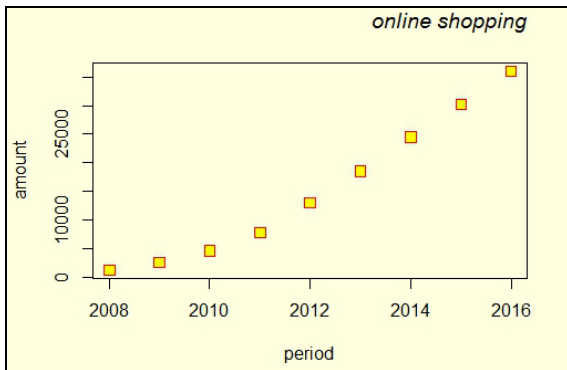


图 4.10 图形美化

4.3.5 交互式绘图命令

交互式绘图，听起来有些“不明觉厉”，实际上交互式绘图命令的操作很简单，其可以帮助用户通过鼠标或键盘等方式与图形的内容进行交互。有时我们需要根据自己的想法而不是数据进行绘图，这就需要非常灵活地在已有图形上做各种修改，R 自身的图形设备有些力不从心，因为要更新图形只能重新绘制。

R 中的交互式绘图函数为此提供了很好的支持，其对于探索性数据分析往往有很大的帮助，它的主要作用有：局部放大、高亮部分元素或调整元素属性（散点图中点的大小、透明度，直方图中的窗宽等）。

R 的交互式函数允许用户直接用鼠标在一个图上提取和提交信息，最简单、最常用的函数是：

```
locator(n, type="n", ...)
```

输入 `locator()` 后，系统等待用户用鼠标左键在图上单击 `n` 个位置，然后返回 `n` 次点击的坐标 `(x, y)`；如果设置了参数 `type`，则在这 `n` 个位置绘制 `type` 指定的图形元素，如绘制符号(`type="p"`)或连线(`type="l"`)。还可以在 `locator` 中设置其他绘图参数，如颜色。比如：

```
> x=rnorm(10) #生成 10 个随机数
> plot(x)      #对 x 绘制散点图
> locator(5, "o", col="red")
$x
[1] 2.621896 4.350554 4.844457 7.762971 7.897672

$y
[1] 0.4210845 0.4483606 -0.9699985 -0.6154088 0.9393311
```

结果如图 4.11 所示。

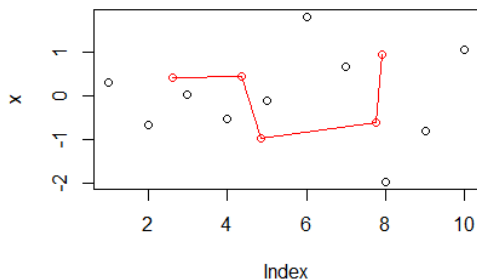


图 4.11 用 `locator()` 进行交互式绘图

在图 4.11 中，输入函数 `locator()` 后用鼠标单击了 5 个位置，在这 5 个位置绘制了红色散点并返回 5 个位置对应的 `(x,y)` 坐标值。另外，不设置 `type` 时 `locator()` 不绘制任何图形元素，只把用户单击的坐标记录下来，这一功能可以用于图形的定位。

```
> locator(2)
$x
[1] 2.666796 2.891297

$y
[1] 1.1575402 -0.3153713
```

使用交互函数会让你觉得非常神奇、灵活。R 中另一个比较有意思的交互函数是 `identify()`，它用于在散点图中找出点。输入它之后，系统读取在图中鼠标按下时指针的坐标位置，然后搜索 `(x,y)` 指定的坐标点，如果这一点足够接近指针的位置，那么将在图中返回指定的图形元素。

```
identify(x, y, labels,...)
```

x 和 y 指定散点图中点的坐标位置,可以是一个向量,向量的长度即要查找的点的个数; $labels$ 是可选的参数,为找到的点添加标签,默认情况下,将在图中添加标签“1”。这个函数看起来有些抽象,实际上使用起来还是很简单的,例如:

```
> plot(x)
> identify(c(4,6),c(1,0),label=c("a","b"),col="red")
warning: no point within 0.25 inches
[1] 1 2
```

绘制结果如图 4.12 所示。

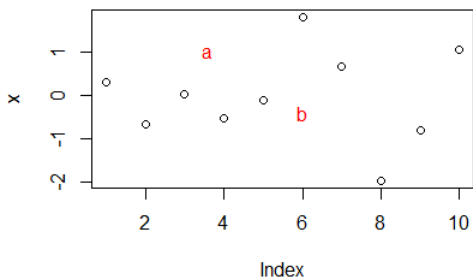


图 4.12 用 `identify()` 进行交互式绘图

输入函数后,用鼠标在点(4,1)附近单击一下,再在点(6,0)附近单击,若是点击的位置偏离较大,则产生 **warning**,警告在指定的坐标位置附近没有找到点,需要重新点击。最后按照已经设置好的标签,分别在两个点处生成图形元素“a”、“b”。

4.4 三维图形

处理多变量的数据集时,二维图形逐渐不能够满足我们的要求,可以通过立体图展示多个变量之间更多的信息。在三个维度的世界里,视觉上的层次分明和色彩上的丰富艳丽,会给人很强的视觉冲击力,留下更深刻的印象。在使用 R 画图时,我们也有必要掌握三维图形的绘制。

R 中绘制三维图形的基本函数有三个,分别为:

- `image(x,y,z)`, 产生长方形的网格,以不同颜色表示 z 的值。
- `contour(x,y,z)`, 以等高线表示 z 的值。
- `persp(x,y,z)`, 产生 3D 表面。

例如 `persp()`, 它是在 X - Y 平面的基础上绘制透视图。

```
persp(x = seq(0, 1, length.out = nrow(z)), y = seq(0, 1, length.out = ncol(z)), z,
      xlim = range(x), ylim = range(y), zlim = range(z, na.rm = TRUE),
      xlab = NULL, ylab = NULL, zlab = NULL, main = NULL, sub = NULL,
      theta = 0, phi = 15, r = sqrt(3), d = 1, scale = TRUE, expand = 1, col = "white")
```

其中，参数 x 和 y 必须是升序的向量，默认情况为等距的从 0 到 1 的数值向量。如果 x 是 list，其成分 $x\$x$ 和 $x\$y$ 将分别用于 x 和 y 。 z 是矩阵，表示要描绘的值（允许 NA）。 θ 和 ϕ 用来定义观察方向的角度。

这里我们将 `demo(persp)` 中的演示示例调用出来加以说明。

```
> x=seq(-10, 10, length.out = 50)
> y=x
#自定义函数 rotsinc
> rotsinc=function(x,y)
+ {
+   sinc=function(x) { y=sin(x)/x ; y[is.na(y)]=1; y }
+   10 * sinc( sqrt(x^2+y^2) )
+ }
> sinc.exp=expression(z == Sinc(sqrt(x^2 + y^2))) #expression 用于转换数学表达式
> z=outer(x, y, rotsinc) #数组外积运算: 通过 x,y 和 rotsinc 函数生成矩阵 z
> oldpar=par(bg = "white")
> persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = "lightblue")
> title(main = sinc.exp)
```

绘制结果如图 4.13 所示。

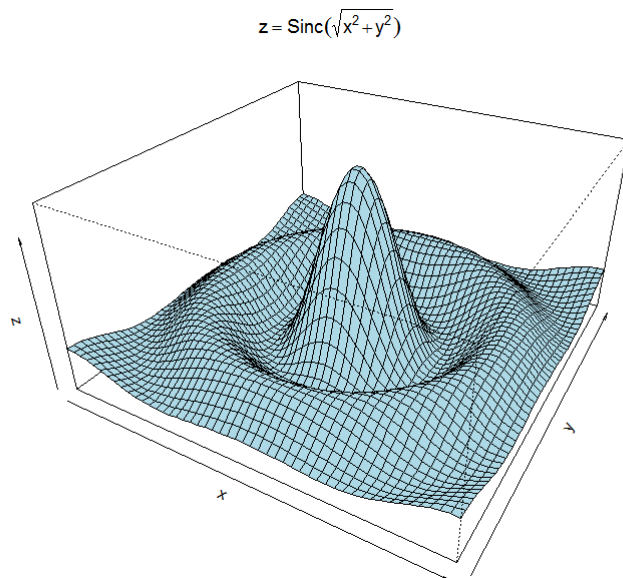


图 4.13 `persp()` 三维绘图

上面提到的三个函数仅适用于绘制比较规范的三维图形，做出效果图。在实际数据分析中，当需要做多个变量的三维散点图时，就需要调用程序包内的函数来完成了。

目前有多个程序包都含有 3D 绘图函数，最常用的是程序包 `rgl` 中的 `plot3d()`，用它绘图的一大好处是，我们可以用鼠标旋转 3D 图，调整到自己希望的角度，这样图形观察起来非常方便。

```
plot3d(x, y, z, xlab, ylab, zlab, type = "p", col, size, lwd, radius, add = FALSE,
       aspect = !add, ...)
```

其中的参数设置与 `plot` 基本类似，`x,y,z` 可以用一个数据框来代替，`size` 表示绘制点的大小。

下面以瑞典死亡率的真实数据集为例，其数据量较大，首先我们需要通过一个三维图形建立一个直观的认识。这里以年份、年龄为 `x, y` 轴，画出死亡率数据的三维散点图。

```
>mortality=read.csv('d:/data/swedish mortality.csv',header=T)
>library(rgl)
>attach(mortality)
The following objects are masked from mortality (position 3):
```

```
Age, Female_death, Female_Exp, L_female_exp, L_male_exp, Male_death,
Male_Exp, q_female, q_male, Year
>plot3d(Year, Age, q_male, col='grey', type='p', zlim=1)
```

绘制结果如图 4.14 所示。

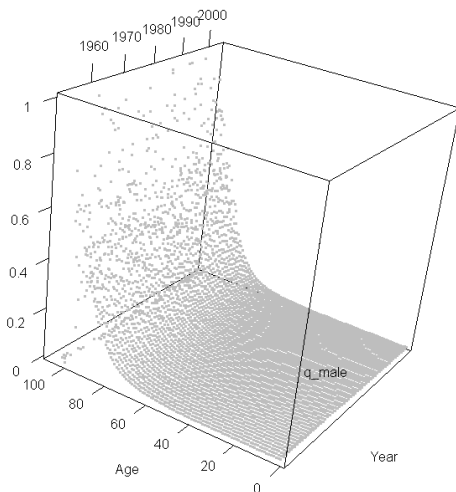


图 4.14 三维散点图

4.5 lattice 程序包

随着实际数据集的复杂程度越来越高，仅仅使用 R 自带的绘图函数已经难以达到理想的图形效果，更多、更强大的绘图程序包就应运而生了。

首先介绍程序包 `lattice`，它主要用于处理结构稍微复杂一些的数据集，它具有入门较容易、

作图速度较快、图形函数种类较多的特点，还可以进行三维绘图。

`lattice` 适用于多个变量的数据集绘图，其中的大部分函数是以一个公式作为主要的自变量，例如 `y~x | z` 表示绘制 `y` 关于 `x` 的图，并以变量 `z` 为分类依据，画出多个图。表 4.8 列出了 `lattice` 程序包中主要的函数。

表 4.8 `lattice` 中的主要函数

函 数	功 能
<code>histogram(~x)</code>	<code>x</code> 的直方图
<code>densityplot(~x)</code>	核密度函数图
<code>qqmath(~x)</code>	理论分布下 <code>x</code> 的分位数图
<code>qq(y~x)</code>	两样本的分位数图， <code>x</code> 是数值型变量， <code>y</code> 可以是数字、字符或因子
<code>stripplot(y~x)</code>	条形图， <code>x</code> 必须是数值， <code>y</code> 可以是因子
<code>bwplot(y~x)</code>	箱线图
<code>dotplot(y~x)</code>	cleveland 点图
<code>barchart(y~x)</code>	<code>y</code> 关于 <code>x</code> 的直方图
<code>xyplot(y~x)</code>	二元图
<code>splom(~x)</code>	矩阵二元图
<code>contourplot(z~x*y)</code>	<code>z</code> 在 <code>x</code> , <code>y</code> 坐标轴下的表面图
<code>levelplot(z~x*y)</code>	同上， <code>z</code> 的彩色图
<code>wireframe(z~x*y)</code>	3D 透视图
<code>cloud(z~x*y)</code>	3D 散点图
<code>parallel(~x)</code>	平行坐标图

之所以说 `lattice` 包很容易入门，是因为其中的函数使用起来非常简单，与 `R` 中基本的函数并无太大差异。与 `plot()` 函数类似，`lattice` 中的各绘图函数也有 `main`、`xlab` 等基本参数设置，可以通过帮助页面查看。例如 `groups` 指定分类变量；`type` 指定图形类型；选项 `auto.key` 在图形中添加图例，图例自动生成，我们只需给出图例所在的坐标位置即可。

下面用一个实例来说明。在程序包 `ggplot2`（下一节介绍）中带有很多结构复杂的数据集，我们将使用其中的 `diamonds` 作图，以给读者展现更好的图形效果。数据集 `diamonds` 包括了钻石的重量（`carat`）、颜色、价格和质量信息，共有 10 个变量，部分数据如表 4.9 所示。

表 4.9 `diamonds` 数据集

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31

续表

	carat	cut	color	clarity	depth	table	price	x	y	z
4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

```
>install.packages("ggplot2")
>library("ggplot2")
>data(diamonds,package="ggplot2")
```

首先, 绘制散点图是最基本的功能, 通过函数 `xyplot()` 作价格和重量的二维散点图, 并根据变量 `cut` 区分散点。这里注意函数内部一些重要参数的含义。`type` 指定图形类型; `p` 表示画出散点; `smooth` 表示画出平滑曲线, 且其光滑程度由参数 `span` 控制; 选项 `auto.key` 在图形中添加图例, `corner=c(1,0)` 表示图例放在右下角; `main` 添加图形标题。

```
># 为了图形效果更好, 从数据中随机抽取 1000 个样本作图
>sample=diamonds[sample(nrow(diamonds),1000),]
>xyplot(price~carat,data=sample,groups=cut,auto.key=list(corner=c(1,0)),type=c("p",
"smooth"),span=0.7,main="Price VS. Carat")
```

绘制结果如图 4.15 所示。

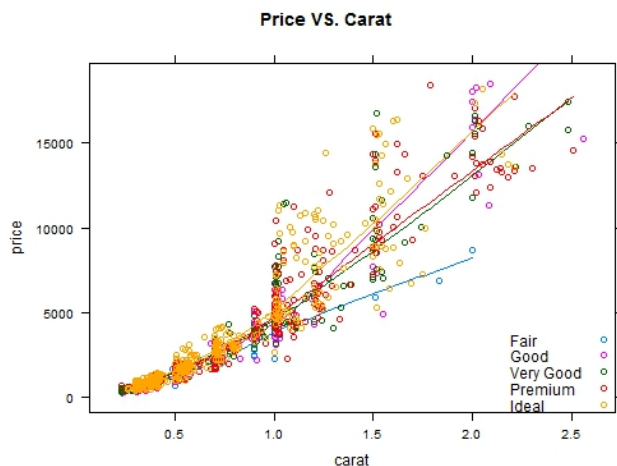


图 4.15 lattice 散点图和平滑曲线

绘制这样一个较为复杂的多元散点图, R 的基本函数也可以完成, 但使用起来会比较麻烦, 因而这也体现了 `lattice` 处理多元数据的便捷性。再比如, 以 `color`、`cut` 分组绘制 `price` 的箱线图, 不需要对绘图区域进行分割, 一个简单的指令就可以完成。

```
>bwplot(color~price | cut,data=diamonds,main="Box-and-Whisker Plots of Price")
```

绘制结果如图 4.16 所示。



图 4.16 lattice 绘制分组箱线图

当然，为了更好地按某一分类变量去比较数据，有些时候也需要分割图形区域。使用 `lattice` 绘图时，分割绘图区域的操作变得很简单，只要设置参数 `layout` 即可。例如以钻石颜色 `color` 为分类变量，使用函数 `histogram()` 绘制价格的直方图，其中参数 `layout` 表示图形的布局，例如 `layout=c(2,5)` 表示图形的排列方式是两列五行。

```
>library(lattice)
>histogram(~price | color,data=diamonds,layout=c(2,4))
```

绘制结果如图 4.17 所示。

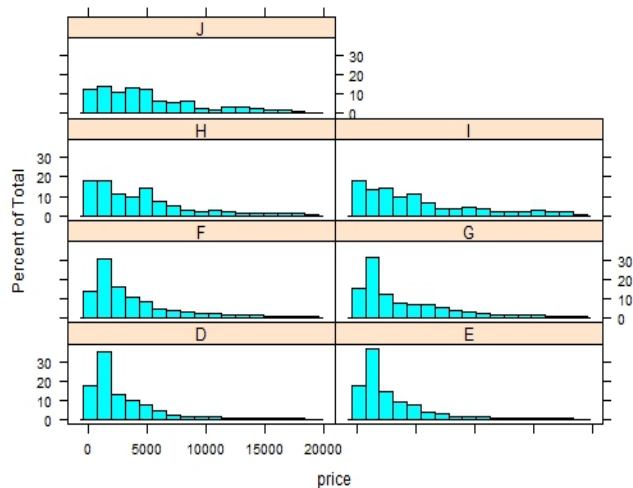


图 4.17 lattice 绘制分组直方图

上文中还提到, `lattice` 中含有绘制三维图形的函数, 其中 `cloud()` 用于绘制三维散点图, 与 `plot3d()` 效果相似, 但可以进行分组绘图; `wireframe()` 用于绘制 3D 表面图, 它与基础包中的 `persp()` 效果相似。下面就用一个绘制 3D 表面图的例子加以说明。

```
> x=seq(-pi,pi,len = 20)
> y=seq(-pi,pi,len = 20)
> g=expand.grid(x=x,y=y) #构造一个数据框, 内部变量为 x,y
> g$z=sin(sqrt(g$x^2+g$y^2)) #对数据框 g 添加变量 z
> wireframe(z~x*y,data=g,draper=TRUE,aspect=c(3,1),colorkey=TRUE,main=expression(z=
sin(sqrt(x^2+y^2))))
```

`wireframe()` 中的参数 `draper` 是一个逻辑值, 表示是否在图形表面着色, 若为 `TRUE`, 3D 表面的颜色将根据 z 轴的不同取值而变化; 若为 `FALSE`, 表面的颜色即图形背景色。`aspect` 是一个长度为 2 的数字向量, 用来确定面板的长宽比, 向量中的两个数值分别表示图 4.18 中封闭立方体的 $y\text{-size}/x\text{-size}$ 和 $z\text{-size}/x\text{-size}$ 两个比值。`colorkey` 为逻辑值, 表示图形旁边是否用列表描述颜色。参数 `main` 设置标题, 通过 `expression()` 对数学公式进行转换。

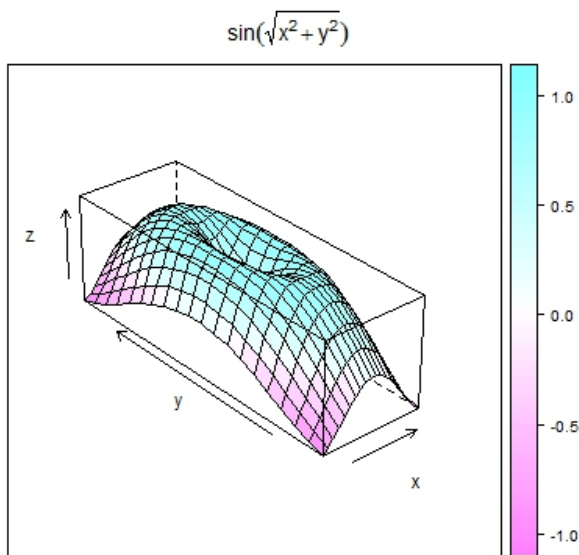


图 4.18 `lattice` 绘制 3D 表面图

此外, 程序包 `scatterplot3d` 中的函数 `scatterplot3d()`、`lattice` 中的函数 `wireframe()` 等也都可以实现类似的作图, 这里不再一一介绍了。

4.6 ggplot2 程序包

R 自带的作图函数还是比较基础的, 功能相对单一, 随着越来越多新程序包的开发, R 语言

的绘图功能日趋强大，图形的美感也直线上升。如果读者可以用 R 绘制出与众不同、数学效果与艺术美感兼备的图形，那么在报告中都能给人留下耳目一新、印象深刻的感觉。接下来，我们介绍一个非常热门的作图工具——**ggplot2**，目前在使用的广泛程度上，它已经远远超过了 **lattice**。

ggplot2 是 R 中用于绘图的高级程序包，它将绘图视为一种映射——数学空间到图形元素空间的映射，例如将不同的数值映射为不同的颜色或其他图形属性。使用过 **photoshop** 的读者对图层的概念一定不陌生，**ggplot2** 在画图时就是采用了图层的设计方式，允许用户一步步构建图形，并且便于图层的修改。**ggplot2** 绘图的另一大好处是简洁、美观，通过控制图层我们可以得到非常理想的图形效果。

4.6.1 快速绘图

首先介绍函数 **qplot()**，它与 **plot()** 的设计类似，适用于快速绘图，所以初学者较容易掌握。它的参数设置与 **plot()** 也比较接近，如 **x**, **y** 指定横、纵坐标的变量，**data** 指定数据集。比较特殊的参数是 **facets**（位图），很多时候我们要将数据按某种方式分组，分别进行绘图，我们就是用 **facets** 来控制分组的方法和排列形式的。**qplot()** 的调用格式如下：

```
qplot(x, y = NULL, ..., data, facets = NULL, margins = FALSE, geom = "auto", stat =
list(NULL),
position = list(NULL), xlim = c(NA, NA), ylim = c(NA, NA), log = "", main = NULL,
xlab = deparse(substitute(x)), ylab = deparse(substitute(y)), asp = NA)
```

下面仍以 **diamonds** 数据集为例，在具体应用中详细说明参数的用法。使用函数 **plot()** 绘制多个变量的图形时，我们需要自己设置图形样式、颜色并添加图例，而 **qplot()** 函数可以自动地为我们完成这些任务，只需用 **shape**、**color** 和 **size** 等参数指定分类变量，R 就可以自动设置图形的形状、颜色和大小。如从原始数据集 **diamonds** 中随机抽取一部分样本绘制价格与重量之间的散点图，根据变量 **cut** 和 **color** 分组。

```
>sample=diamonds[sample(nrow(diamonds),200),]
#参数 shape 指定变量 cut 为分类依据，按变量 color 绘制不同颜色的散点
>qplot(carat,price,data=sample,shape=cut,color=color)
```

绘制结果如图 4.19 所示。

qplot() 函数并不局限于绘制散点图，通过改变参数 **geom**，可以绘制多种图形样式。

- **geom="points"** 绘制散点图，当 **qplot(x,y)** 包含 **x,y** 两个参数时，默认绘制散点图。
- **geom="smooth"** 表示对所有散点拟合一条曲线。
- **geom="boxplot"** 生成数据点分布的箱线图。
- **geom="path"** 和 **geom="line"** 在数据点之间连线，常用于绘制时间序列图。

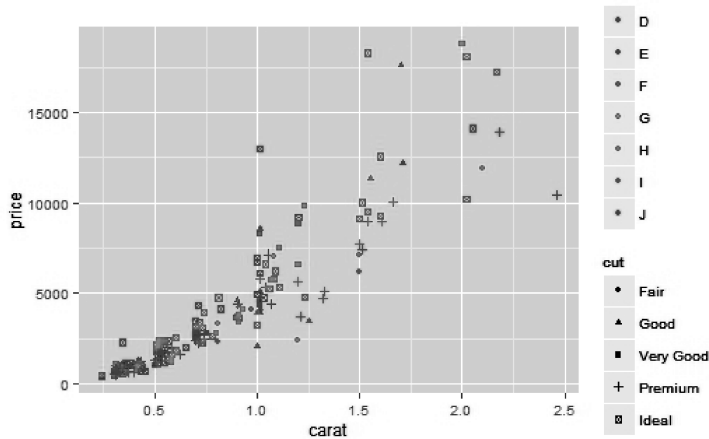


图 4.19 qplot()绘制散点图及参数设置

另外，对于连续变量而言，

- `geom="histogram"` 绘制直方图，当 `qplot(x)` 只含有 `x` 这一个参数时，默认绘制直方图。
- `geom="freqpoly"` 绘制频数多边形。
- `geom="density"` 绘制密度函数图。

离散型变量通常用 `geom="bar"` 绘制条形图。

例如，在上述散点图中添加一条平滑曲线，通过 `method` 参数可以指定曲线拟合的方法，默认为 `method="loess"`——平滑局部回归。参数 `span` 控制曲线的平滑程度，取值越大曲线越平滑。

```
>qplot(carat,price,data=sample,geom=c("point","smooth"),span=0.3)
geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use
'method = x' to change the smoothing method.
```

绘制结果如图 4.20 所示。

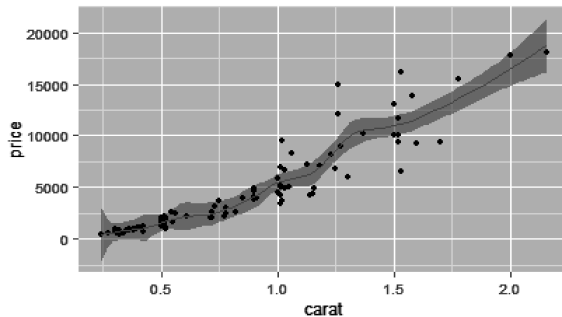


图 4.20 散点图及平滑曲线

我们已经强调过直方图在统计分析中的重要作用，下面使用 `qplot()` 对变量 `carat` 画出更美观的

直方图。其中参数 `binwidth` 用于调整柱形的宽度，取值越大则柱形越宽。需要特别注意的一点是，在设置直方图颜色时，要使用参数 `fill` 填充，而不是 `color`（`color` 仅改变柱形外边框的颜色）。

```
>qplot(carat,data=diamonds,geom="histogram",binwidth=0.1,xlim=c(0,3),fill=color)
```

绘制结果如图 4.21 所示。

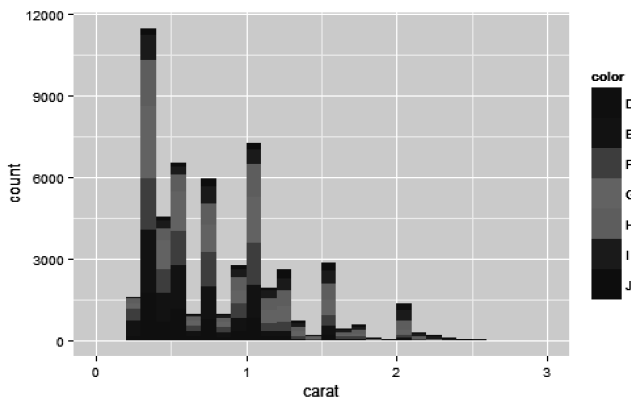


图 4.21 qplot()绘制直方图

4.6.2 分图层绘图

`qplot()`与 `plot()`函数的使用方式比较类似，而 `ggplot()`真正引入了图层的概念，它可以更灵活地绘图和修改。图层允许用户一步一步地构建图形，程序包中包含很多类函数，例如数据、映射、几何对象、统计变换等，每类函数都可作为一个单独的图层，下面我们来分门别类地介绍。

（1）数据和映射

在加载程序包后，绘图的第一个步骤是用 `ggplot` 定义第一个图层，第一个图层将数据中的变量通过 `Mapping` 映射到图形属性。

```
ggplot(data, mapping=aes(x, y, <other aesthetics>))
```

其中，`data` 指定数据集；参数 `mapping` 用于构建映射，通常使用函数 `aes()`来指定 `x` 轴和 `y` 轴的变量，还可以指定其他分类变量，如颜色（`color`）、形状（`shape`）和大小（`size`）等。

这里仍使用数据集 `diamonds` 为例，随机选取其中的 1000 个样本绘图。首先使用 `ggplot()`构建第一图层，确定数据来源，指定 `carat` 为 `x` 轴变量、`price` 为 `y` 轴变量，并将变量 `clarity` 映射为颜色属性。第一图层是对图形的初始化，只定义了“映射”，所以在绘图区域中没有图形，我们可以将定义的第一图层存储于 `p` 中。

```
>sample=diamonds[sample(nrow(diamonds),1000),]
> p=ggplot(data=sample,mapping=aes(x=carat,y=price,color=clarity))
```

(2) 几何对象

基本图层确定了数据源和映射后，通过加号(+)就可以不断地添加新图层，一般的步骤是，第二图层添加几何对象类的函数，在图中绘制图形元素，如点、线、多边形等，还可以用来绘制其他类型的图形，如直方图、箱线图等等。

几何对象类别的函数以“geom_”开头，后面接所需的图形样式，与 `qplot()` 中的参数 `geom` 是一致的。表 4.10 列出了比较常用的一些函数。

表 4.10 几何对象 (geom) 函数

主要函数	功 能
<code>geom_abline()</code>	绘制直线，通过参数 <code>intercept</code> 、 <code>slope</code> 定义
<code>geom_bar()</code>	条形图
<code>geom_blank</code>	空白，只显示横纵坐标，不绘制任何图形；在此基础上可以继续添加图形对象
<code>geom_bin2d()</code>	二维热图
<code>geom_contour()</code>	三维表面图
<code>geom_densitiy()</code>	平滑密度曲线
<code>geom_dotplot()</code>	点图
<code>geom_histogram()</code>	直方图
<code>geom_hline()</code> / <code>geom_vline()</code>	水平线 / 垂直线
<code>geom_jitter()</code>	增加扰动
<code>geom_line()</code>	曲线图
<code>geom_map()</code>	数据地图
<code>geom_path()</code>	路径图，按原始顺序连接所有观测变量
<code>geom_point()</code>	散点图
<code>geom_polygon()</code>	多边形
<code>geom_smooth()</code>	平滑曲线，添加平滑的条件均值
<code>geom_text()</code>	添加文本注释

上面函数内部的基本参数都是一样的。以散点图为例：

```
geom_point(mapping = NULL, data = NULL, stat = "identity", position = "identity", na.rm = FALSE, ...)
```

参数 `mapping` 用于构建映射，`data` 指定数据集，如果在第一图层已经指定，则可以省略；`stat` 用于这一层数据的统计变换；`position` 用于这一层图形的位置调整，常用于条形图 (`bar`) 和直方图，取值为“`identity`”时表示直接显示，“`dodge`”为按分类变量并列放置，“`stack`”为堆叠放置，“`fill`”显示相对比例；“`jitter`”为增加扰动，常用于散点图，防止图形过分重叠。

接上例，使用“+”添加第二图层散点图和第三层的平滑曲线，这时绘图区域将显示出我们

定义的几何图形。

```
>p+geom_point()+geom_smooth()
geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use
'method = x' to change the smoothing method.
```

与上面代码等价的另一种写法是:

```
> d=ggplot(data=sample,aes(x=carat,y=price,color=clarity))+geom_point()+geom_smooth()
> print(d) #绘制的命令
```

绘制结果如图 4.22 所示。

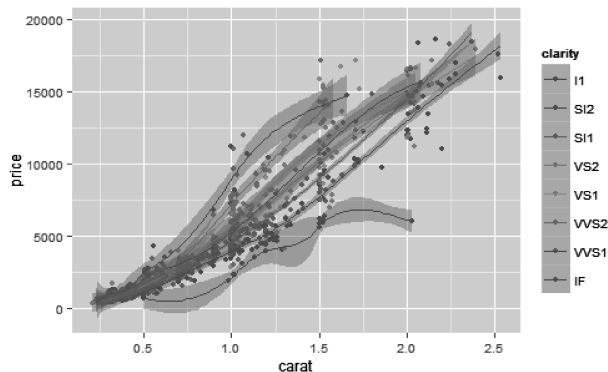


图 4.22 散点图和平滑曲线

上图在第二个图层 `geom_point()` 中没有设置参数, 则根据第一层的参数设置, 以系统默认的方式绘图, 即 `clarity` 为分类变量, 对数据分别进行平滑。但是, 如果在 `geom_point()` 中设置了参数, 那么其反映的属性仅针对这一层的图形, 不会对其他图层产生影响。

例如, 我们想要对上面的图形进行整体平滑, 可以将参数 `color` 设置在第二图层 `geom_point()` 中, 这样第三图层的平滑曲线不会再根据 `clarity` 分类绘制。

```
> p=ggplot(data=sample,aes(x=carat,y=price)) #这里不再指定 color 的分类变量
>p+geom_point(aes(color=clarity))+geom_smooth() #仅第二图层分类画点
geom_smooth: method="auto" and size of largest group is >=1000, so using gam with
formula: y ~ s(x, bs = "cs"). Use 'method = x' to change the smoothing method.
```

绘制结果如图 4.23 所示。

进行数据映射时, 函数 `aes()` 可用于设置图形样式, 通过参数 `color`、`shape` 和 `size` 分别设置点的颜色、形状和大小按哪些向量分类, 通过这些参数, 即使一个简单的散点图也可以传递大量信息。

```
> sample=diamonds[sample(nrow(diamonds),100),] #从 diamonds 中取一个样本量为 100 的子集
> p=ggplot(data=sample,aes(x=carat,y=price))
> #设置三个分类变量 color、cut 和 clarity, 分别用不同颜色、形状和大小表示
```



```
>#参数 alpha 控制透明度
```

```
>#position="jitter"对散点增加扰动,防止点的过度重叠
```

```
> p+geom_point(aes(color=color,shape=cut,size=clarity),alpha=0.5,position="jitter")
```

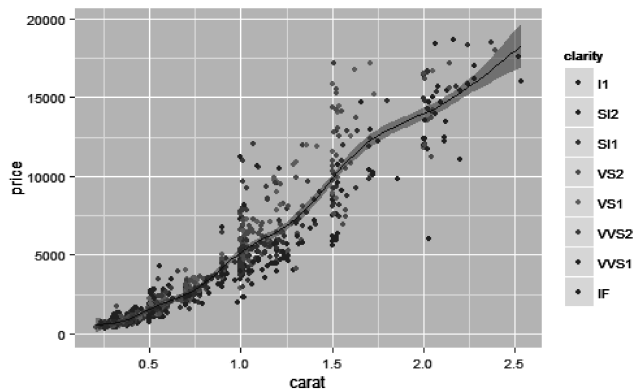


图 4.23 整体平滑

绘制结果如图 4.24 所示。

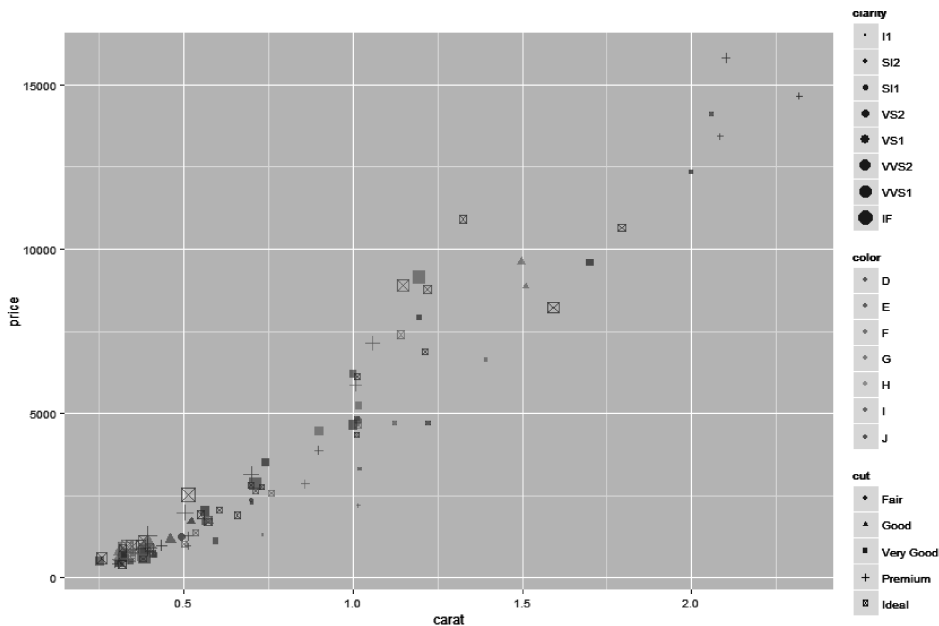


图 4.24 图形样式的设置

(3) 标度

标度负责控制图形属性的显示方式,主要包括设置坐标轴刻度,修改颜色取值、图例样式等。

使用标度类的函数，相当于添加一个新的图层，因此仍然用“+”连接函数，除了基本图层 `ggplot()`，其他图层的设置都可以应用于函数 `qplot()`。

首先，设置坐标轴样式的标度函数一般以“`scale_x`”开头，如 `scale_x_date` 显示日期标度，`scale_x_discrete` 设置离散型标度，最常用的是函数 `scale_x_continuous()`，其用于设置连续性刻度，它们的内部参数都是一致的，功能如表 4.11 所示。

表 4.11 `scale_x_continuous()` 的参数设置

参 数	含 义
<code>names</code>	更改轴标签
<code>limits</code>	设置坐标轴范围
<code>breaks</code>	设置坐标的刻度线
<code>labels</code>	手动标记刻度的名称
<code>trans</code>	用于对坐标轴进行数学变换

有三个内置的坐标轴转换函数可以直接使用，`scale_y_log10()` 表示对 `y` 轴作 \log_{10} 转换；`scale_y_sqrt()` 对坐标轴作开放转换；`scale_y_reverse()` 将 `y` 轴刻度反转。

此外，标度类的函数中还包括一些短函数，如控制坐标轴界限的 `xlim()`、`ylim()`，修改坐标刻度的 `labs(x="", y="")`。

例如绘制重量（`carat`）的直方图，并控制 `x` 轴的范围，可以使用 `scale_x_continuous (limits)`，也可以使用短函数 `xlim()`。

```
> p=ggplot(data=diamonds,aes(x=carat)) #指定x轴为carat变量
> p+geom_histogram()+scale_x_continuous(limits=c(0,3))+opts(title="Hist of carat")
> #可以使用xlim(0,3)代替 scale_x_continuous(limits=c(0,3))
> #opts(title)添加图形标题
stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```

绘制结果如图 4.25 所示。

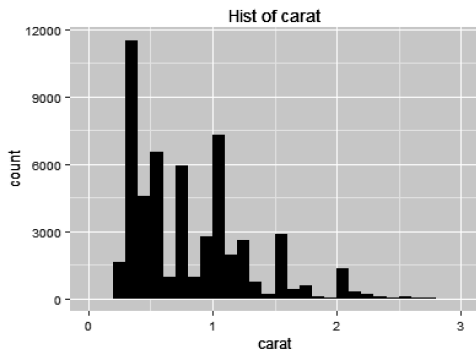


图 4.25 坐标轴设置

接下来，修改图形样式的函数有：`scale_manual` (`scale_alpha_manual`, `scale_color_manual`, `scale_shape_manual`, `scale_size_manual`)对透明度、颜色、形状的离散取值作映射，函数 `scale_shape`、`scale_size` 以及 `scale_linetype` 专门用于修改形状、大小和线型，可以作连续或离散取值的映射。`ggplot2` 中修改图形颜色的标度函数最多，它们都以“`scale_color`”开头，例如 `scale_color_gradient`、`scale_color_grey` 等。

在上面的直方图中，如果想要对 `color` 的不同取值，赋予特定的颜色，可使用指令 `scale_color_manual(values)` 完成，其中 `values` 为字符串向量，例如 `c("blue","red",...)` 可自定义颜色。

```
> p=ggplot(data=diamonds,aes(x=carat,fill=color))
> #注意，由于要绘制直方图，color 应该映射到参数 fill 表示填充颜色
> p+geom_histogram()+xlim(0,3)+scale_colour_manual(values = rainbow(7)) #定义为彩虹色
```

绘制结果如图 4.26 所示。

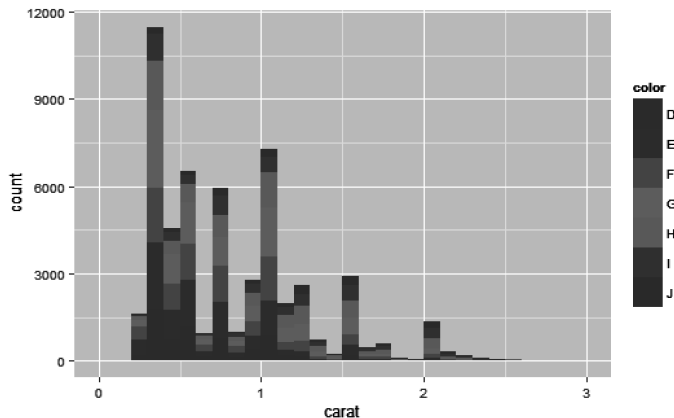


图 4.26 修改图形颜色

(4) 统计变换

统计变换函数以“`stat`”开头，它们可以对原始数据进行某种函数变换，是非常重要的功能。我们可以自定义函数，基于原始数据计算并在图上表现出来，也可以通过它们改变“`geom_`”函数画图的默认统计参数。

`stat` 函数很多，举几个比较常用的例子：`stat_ecdf` 在原始数据基础上计算经验累积分布函数并画出曲线图，`stat_density` 绘制核密度估计曲线图，`stat_qq` 绘制 qq 图，`stat_smooth` 添加平滑曲线，`stat_unique` 删除重复值，等等，函数 `stat_function` 还可以自定义一个函数并绘出图形。事实上，每一个几何对象都内置了一个统计变换。比如散点图的几何对象是点，内置统计变换为 `stat_identity`（相等）；直方图的几何对象是条形，内置变换为 `stat_bin`——按照区间统计频数。

例如用 `stat_smooth` 对数据作 `loess` 平滑，在 `carat-price` 散点图上添加非线性回归线。

```
> sample=diamonds[sample(nrow(diamonds),1000),]
> ggplot(sample,aes(x=carat,y=price))+geom_point()+scale_y_log10()+stat_smooth()
> #第二图层添加散点；第三图层对 y 轴作 log10 变换；第四图层添加平滑的统计变换
```

绘制结果如图 4.27 所示。

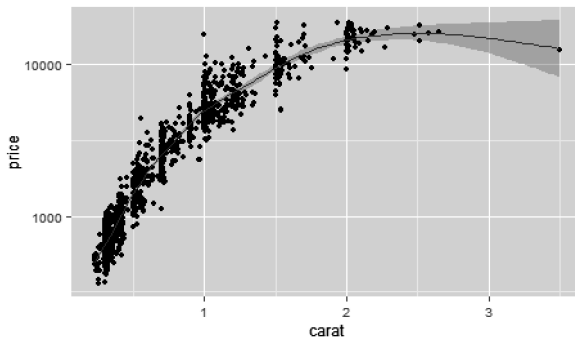


图 4.27 统计变换

(5) 分面

当我们想要观察某一分类变量对数据的影响情况时，仅通过 `shape`、`color` 区分是不够的，往往需要根据变量的不同取值进行分组、分别绘图。这时就要用到 `facet` 函数，它控制数据分组的方法和排列形式，进行条件绘图。

常用的函数是 `facet_wrap(~x, ncol)`，其中 `x` 表示分组变量，`ncol` 表示图形的排列方式，即分成几列。也可以用 `facet_grid(x~.)` 替代。例如图 4.28 所示的平滑曲线，若按变量 `cut` 分组绘制，图形效果会非常混乱，但利用分面功能，就可以分别作图，便于比较。

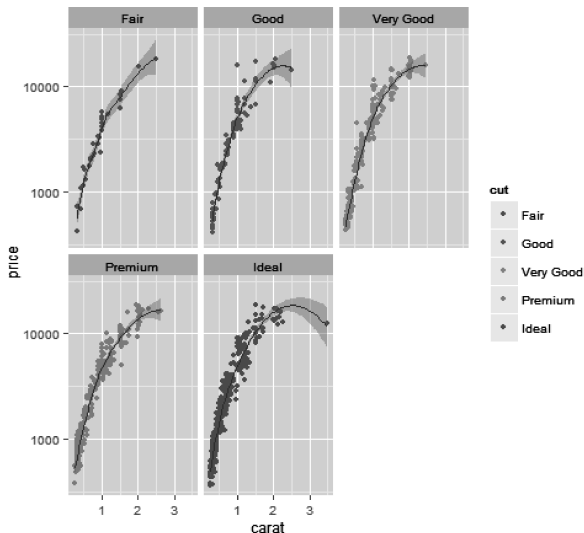


图 4.28 分面绘图

```
>ggplot(sample,aes(x=carat,y=price))+geom_point(aes(colour = cut))+
+ scale_y_log10()+stat_smooth()+facet_wrap(~cut,ncol=3)
>#在 geom_point() 中将 cut 映射到颜色属性中; facet_wrap() 中的 ncol=3 表示每行画 3 个图
```

(6) 坐标系

坐标系可以对坐标轴进行某种变化以满足不同的绘图需求，主要的函数总结在表 4.12 中。

表 4.12 坐标系函数

函 数	功 能
coord_flip()	坐标轴翻转：x、y 轴互换
coord_polar()	转换成极坐标（常用于绘制饼图、靶心图、风玫瑰图等）
coord_cartesian()	笛卡儿坐标系
coord_map()	将坐标系投影于程序包 <code>mapproj</code> 中提供的地图
exp_trans()	坐标轴变换，如对数、指数变化

介绍了这么多 `ggplot2` 的绘图方法，细心的读者可能发现，一直没有提及饼图，事实上，在 `ggplot2` 中饼图就是柱状图，只不过是使用极坐标而已，条形图的高度对应为饼图的角度。

例如，按不同的切工（变量 `cut`）分组绘制饼图。第一图层定义数据集为 `diamonds`；第二图层绘制条形图，为了便于绘制饼图将横坐标设为因子“1”；最后，第三图层对坐标轴作极坐标变换，就完成了饼图的绘制。

```
>ggplot(diamonds)+geom_bar(aes(x=factor(1),fill=cut))+coord_polar(theta="y")
>#theta="y"表示将条形图 y 轴高度转换为饼图的角度
```

绘制结果如图 4.29 所示。

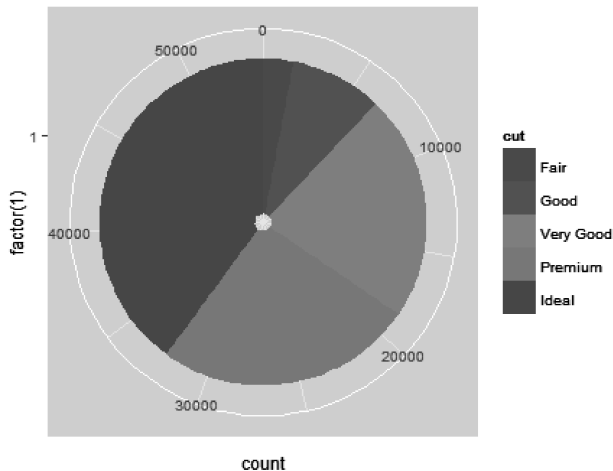


图 4.29 饼图（通过坐标轴变换）

4.7 图形保存

完成绘图后，最后一步是按照指定文件格式、属性保存和导出图形，以备以后使用。R 绘制好的图可以保存成多种格式，对应的生成函数名即它的扩展名。可生成的文件格式有 png、jpeg 和 pdf：

```
png(file="myplot.png", bg="transparent")
jpeg(file="myplot.jpeg")
pdf(file="myplot.pdf")
```

生成文件后，默认在后台打开，所以查看图形文件前需要用 `dev.off()` 关闭文件。例如，将上面的饼图保存为 .png 文件，输入下面指令后就在 D 盘的文件夹 data 中生成了图形文件。

```
>png(file="d:/data/pie.png", bg="transparent")
>dev.off()
```

此外，程序包 `ggplot2` 中的函数 `ggsave()` 也用于保存图形，并且可以指定为不同的文件类型。

```
ggsave(filename = default_name(plot), plot = last_plot(),
device = default_device(filename), path = NULL, scale = 1, ...)
```

`filename` 指定生成文件的路径、名称及扩展名，文件路径也可以通过 `path` 设置；`plot` 填写图形对象，默认为最后显示的图形；`device` 指定要使用的设备，自动提取文件扩展名；`scale` 为比例因子。将上面的饼图保存成一个 pdf 文件，只需要一条简单的指令就可以完成。

```
>ggsave(filename="d:/data/pie.pdf")
Saving 7.67 x 4.53 in image
```

这样就生成了一个 pdf 文件，还可把图形保存成 .png 格式。

4.8 实战案例：数据地图

在实际工作中经常碰到这种情况，收集到的数据与地理位置有关，例如全国各省市的气温、收入水平等，如果能将数据和地图结合起来，那么将收到很好的视觉效果。

数据地图是一种经典的图示方法，因此我们有必要掌握如何在 R 软件中实现数据地图的绘制。前面已经详细介绍了程序包 `ggplot2` 的用法，其中函数 `geom_map()` 在绘制数据地图时非常好用。

```
geom_map(mapping = NULL, data = NULL, map, stat = "identity", ...)
```

参数 `mapping` 用于构建映射，通常用函数 `aes()` 指定数据集的变量；数据集则存放在 `data` 中；`map` 是一个数据框，包含地图的所有基本信息，除了 R 中自带的美国地图外，如果我们要绘制其他国家的数据地图，通常使用 `fortify()` 创建一个空间对象，其中必须包括列 `x` 或 `long`、`y` 或 `lat`、`region` 或 `id` 这三个变量，用于指定各区域的坐标位置。`stat` 表示这一层数据的统计变换，默认为“identity”不作任何变换，即直接使用原始数据。

举一个有趣的例子，NBA 比赛的球队一共有 30 支，现在我们想了解“小皇帝”詹姆斯（LBJ）对阵哪个球队时表现最好，或者说得分更高呢？这时绘制一个数据地图，就可以得到很直观的结论。

首先，我们要收集原始数据，其中包含 3 个变量：Opp 表示对手的球队名称；AvgPTS 为詹姆斯对阵该队时的平均得分；最重要的变量是 state，表示球队所在的州，各州的名称必须与 R 中内置的美国地图信息保持一致，美国地图的数据信息通过函数 map_data() 获取。另外，由于 NBA 球队并不是遍布每一个州，对于没有球队的州，我们将对应的 AvgPTS 赋值为 0，这样绘制图形时就会以灰色区域表示了。

```
>library(ggplot2)
> lbj=read.table("d:/data/lbj.txt",header=T,quote=" ") #读取数据
>attach(lbj)
> head(lbj) #查看数据集的前 5 行
  Opp AvgPTS      state
1 ATL  21.75 georgia
2 BOS  29.25 massachusetts
3 BRK  21.67 new york
4 CHA  30.00 north carolina
5 CHI  28.00    illinois
6 CLE  27.67      ohio
> state_map=map_data("state") #获取美国地图的数据信息
```

接下来开始 ggplot 绘图，定义如下几个图层：

- ① 第 1 图层，ggplot() 指明数据集 lbj，变量 state 为各州名称，即 map_id。
- ② 第 2 图层，geom_map() 绘制地图，将得分 AvgPTS 映射到填充颜色 fill 上，地图信息 state_map 映射到变量 map 上。
- ③ 第 3 图层，expand_limits() 设置横坐标轴的范围。
- ④ 第 4 图层，修改填充颜色和图例样式。红色表示得分高，黄色表示得分较低，颜色的变化范围是 19~max(AvgPTS)；guide="colorbar" 表示图例为颜色条。
- ⑤ 第 5 图层，添加图形标题。
- ⑥ 第 6 图层，在图形上添加各球队名称。

```
>p=ggplot(lbj,aes(map_id=state))+geom_map(aes(fill=AvgPTS),map=state_map)+
+ expand_limits(x=state_map$long,y=state_map$lat)+
+
+ scale_fill_continuous(limits=c(19,max(AvgPTS)),high='red3',low='yellow',guide="colorbar")+
+ opts(title='詹姆斯客场平均得分')
```

使用 ggplot() 绘制地图的指令其实并不难，但是添加第 6 图层——各球队名称需要花费一番功夫了。首先我们要获取各州的坐标位置，分别存放在向量 xx 和 yy 中：

```
> attach(state_map)
```

```

> state.uni=unique(region) #存放各州的名称
> xx=0;yy=0 #事先建立变量xx和yy,下面用循环找到每个州对应的坐标值
> for(i in 1:length(state.uni)) {
+ xx[i]=mean(long[region==state.uni[i]])
+ yy[i]=mean(lat[region==state.uni[i]])
+ }

```

接下来使用 `ggplot2` 包中的函数 `annotate()` 在图中添加标签。`annotate(geom,x,y,...)` 专门用于在图中添加注释,注释类型由参数 `geom` 指定,由于我们要添加的是文本,所以设 `geom="text"`。在此之前,还要找到各州对应的球队名称,作为要添加的“标签”。

```

> order=0 #按变量state.uni的顺序找到数据集lbj中各州的位置,存放于变量order
> for(i in 1:length(state.uni)) {
+ order[i]=which(state==state.uni[i])
+ }
> labels=Opp[order] #通过位置找到各州对应的球队名称
> p+annotate("text",x=xx,y=yy,label=labels) #最后绘图并添加注释

```

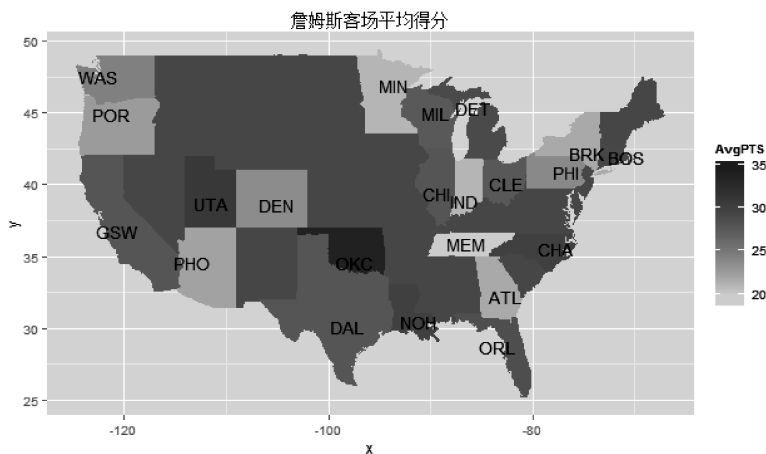


图 4.30 美国数据地图：LBJ 得分

图 4.30 中的区域颜色越红（看上去颜色越深），表示詹姆斯在该球队所得的分数越高，状态比较火热；黄色代表得分较低。可见，詹姆斯在面对他的劲敌 OKC（俄克拉荷马雷霆队）时，会打出很好的水平，得分也较高。

可以发现，利用 `ggplot2` 绘制美国地图是比较容易的，因为美国地图是程序包 `maps` 中自带的地图。但其他国家的地图数据则要从外部导入，推荐从 `GADM` 获取全球各行政区域的数据，`GADM` 是世界行政区域（或行政区域界线）位置的空间数据库，可专门用于地理信息系统和类似的软件，并且提供了 `Rdata` 格式的数据，非常便于在 `R` 中绘制数据地图。

在实际调查项目中，对同一个指标，我们经常能够获取全国各地区的数据，例如从国家统计

局可以获取 2010 年第六次人口普查我国各地区的人口数据，如果我们在 R 中把人口信息绘制在中国地图上，即根据人口的多少对各省份进行着色，就需要从 GADM 导入地图信息。

为了把网站上获得的地图信息转换成 ggplot2 需要的数据框格式，要用到程序包 gpclib，但是在 R3.0.1 版中，这个程序包目前还无法安装使用，因此我们介绍 R 中另一种绘制地图的方法，即使用空间分析程序包 sp()，使用其中的函数 spplot() 绘制数据地图，比 ggplot2 的方法更简单。

```
>install.packages("sp")
> library(sp) #安装并加载程序包
> #从 gadm.org 网站上得到中国的省区地理数据，并加载到 R 软件内存中
>load(url("http://gadm.org/data/rda/CHN_adm1.RData"))
>#将每个省人口数据按顺序存放在数据框 gadm 中，生成一个变量 pop
> gadm$pop=c(1961,1293,7185,3571,2470,4374,2745,3831,2301,7866,5442,
+           5950,3689,4456,9579,9402,5723,6570,10432,4602,867,2884,
+           8041,3474,4596,300,3732,2557,562,630,2181,706)
> #利用空间绘图命令进行绘图
> spplot(gadm,"pop",col.regions = rev(terrain.colors(gadm$pop)),main="中国各省人口数
据")
```

spplot() 里面第一个参数指定数据集，第二个参数指定存储各省数据的变量名，col.regions 设置颜色的变动范围，如 topo.colors、terrain.colors、rainbow.colors 等。虽然 spplot() 绘图速度较慢，但图形效果很好，在图 4.31 中，颜色越深，表示省内人口越多；颜色越淡，表示人口数越少。

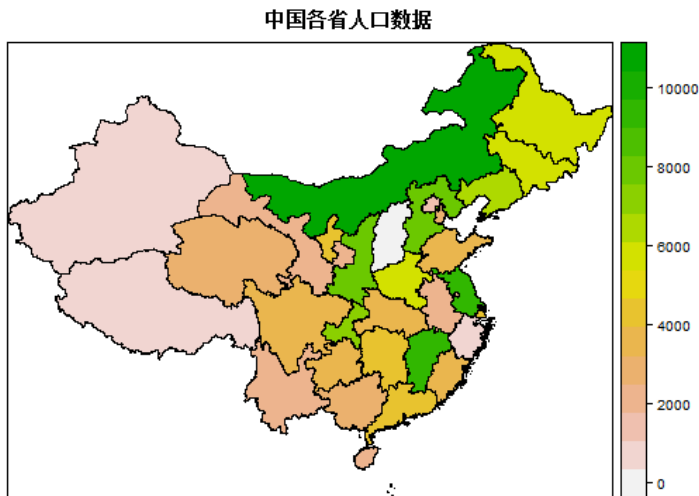


图 4.31 中国数据地图：省人口数

第 5 章

数据的描述性分析

通过前面两章的学习，我们知道，数据收集是取得统计数据的过程，数据预处理是将数据中的问题清理干净，那么接下来的步骤就是统计分析了。数据分析是通过统计方法研究数据的过程，所用的方法分为描述性统计和统计推断两部分。描述性统计用编制图表、计算统计量等形式对数据进行加工处理和显示，进而综合、概括和分析，得出反映所研究数据的一般性特征——数字特征、分布状态等。

描述性统计分析是统计分析的第一步，做好这一步是下面进行正确统计推断的先决条件。

本章将分别从统计量和图形的角度描述样本的分布特征，重点介绍如何运用 R 中的函数完成。

5.1 R 内置的分布

分布是描述一个样本数据最核心、最重要的方式。R 内嵌了很多常用的统计分布，提供了四类函数：概率密度函数（density）、累积分布函数（probability）、分位数（quantile）和伪随机数（random）。在 R 中分别用 d、p、q、r 表示这 4 个项目，后面接分布的英文名称或缩写。

例如正态分布的 4 个函数分别表示为 `dnorm`、`pnorm`、`qnorm` 及 `rnorm`。表 5.1 列出了 R 中的分布函数，每个分布函数都有各自的参数，有些有默认值（如正态分布默认为 $N(0,1)$ ），有些则没有。

最常用的函数一般都来自于统计分析程序包 `stats`，它不需要手动安装，打开 R 的同时其就已经自动加载。另外，有些分布较为复杂但在统计上应用较广，例如 `Wishart` 分布和 `Pareto` 分布，使用它们需要加载新的程序包。

表 5.1 R 内置的分布及对应函数

分 布	R 函数	参数及默认值	所属程序包
贝塔 Beta	_beta	shape1, shape2, ncp=0	stats
二项 Binomial	_binom	size, prob	
柯西 Cauchy	_cauchy	location=0, scale=1	
卡方 Chi-square(χ^2)	_chisq	df, ncp	
指数 Exponential	_exp	rate	
F 分布 Fisher-Snedecor	_f	df1, df2, ncp	
伽马 Gamma	_gamma	shape, scale=1	
几何 Geometric	_geom	prob	
超几何 Hypergeometric	_hyper	m, n, k	
对数正态 Lognormal	_lnorm	meanlog=0, sdlog=1	
逻辑斯蒂 Logistic	_logis	location=0, scale=1	
负二项 Negative binomial	_nbinom	size, prob	
多项式 Multinomial	_multinom	size, prob	
正态 Normal	_norm	mean=0, sd=1	
泊松 Poisson	_pois	lambda	
学生 Student's t	_t	df	
均匀 Uniform	_unif	min=0, max=1	
威布尔 Weibull	_weibull	shape, scale=1	
威尔考克森 Wilcoxon	_wilcox	m, n	
帕累托 Pareto	_pareto	shape, scale	actuar (含有许多逆函数)
布尔 Burr	_burr	shape1, shape2, rate=1 (scale=1/rate)	
逆指数 Inverse Exponential	_invexp	rate	
狄利克雷 Dirichlet	_dirichlet	alpha	MCMCpack (蒙特卡罗, 马氏链)
威沙特 Wishart	_wish	v, S	
逆威沙特 Inverse Wishart	_iwish	v, S	
广义极值 Generalized Extreme Value	_gev	xi, mu, sigma	evir (极值统计分析)
广义帕累托 Generalized Pareto	_gpd	xi=1, mu=0, sigma=1	
多元正态 Multivariate Normal	_mvnorm	mean, sigma	mvtnorm
多元 t 分布 Multivariate-t	_mvt	sigma=diag(2), df=1	mvtnorm

5.2 集中趋势的分析

5.2.1 集中趋势的测度

样本观测值来自于总体，其中含有总体各方面的信息，乍看这些信息有时显得杂乱无章，需要对样本进行加工，通过探索性分析、计算统计量来提炼有用的信息。

描述数据分布特征的统计量可分为两类：一类表示数量的中心位置；另一类表示数量的变异程度（或称离散程度）。两者相互补充，共同反映数据的全貌。

描述统计分布集中趋势的指标主要是平均数、中位数、众数，也称为“平均指标”。这些指标的主要作用包括：

- 反映总体各单位变量分布的集中趋势和一般水平；
- 便于比较同类现象在不同单位之间的水平；
- 便于比较同类现象在不同时期的发展变化趋势或规律；
- 用于分析现象之间的依存关系。

（1）均值（平均数）

均值是最常用于度量数据平均水平的统计量，记为 \bar{x} ，定义为

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

（2）众数

一组数据中出现次数最多的观测值叫做众数，用 M_0 表示。众数测度数据的集中性趋势，一般在数据量较大的情况下，众数比较有意义。众数不受极端值的影响，一组数据分布的最高位置对应的数值即众数。

如果数据没有明显的集中趋势，那么众数可能不存在；如果有两个最高峰点，那么这组数据就有两个众数。

（3）中位数

中位数简单来讲就是数据排序位于中间位置的值，记为 M_e ，即

$$M_e = \begin{cases} x_{(\frac{n+1}{2})} & \text{当 } n \text{ 为奇数时} \\ \frac{1}{2} \left(x_{(\frac{n}{2})} + x_{(\frac{n}{2}+1)} \right) & \text{当 } n \text{ 为偶数时} \end{cases}$$

中位数描述数据中心位置，对于对称分布的数据，均值接近中位数；偏态分布是指频数分布不对称，集中位置偏向一侧，数据的均值则与中位数不同。

中位数是描述分析中重要的统计量，与众数类似，它的显著特点在于不受异常值的影响，具有稳健性。

平均数、众数和中位数三者一起可以对数据的分布形式做简单描述。

- 如果数据具有单一众数且分布对称，那么均值、众数和中位数相等，即 $\bar{x} = M_e = M_0$ 。
- 若集中位置偏向数值小的一侧，称为正偏态分布，也称右偏分布（可以简单地记为分布的“尾巴”在右边）。此时数据存在极大值，必然拉动均值向极大值一方靠拢，因此 $\bar{x} > M_e > M_0$ ，如图 5.1(b)所示。
- 如果集中位置偏向数值大的一侧，称为负偏态分布，也叫左偏分布，即 $\bar{x} < M_e < M_0$ ，如图 5.1(a)所示。

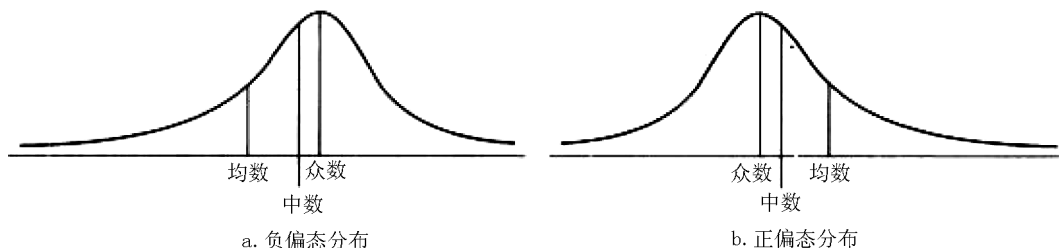


图 5.1 偏态分布

(4) 分位数

中位数是将数据等分后中间位置的点，与之类似的还有四分位点、十分位点和百分位点。常用的是数据排序后处于 25% 和 75% 位置上的值，即四分位点。

5.2.2 R 语言实现

R 中的函数 `mean()` 用于计算样本均值，其调用格式如下：

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

内部的参数 `x` 是计算对象，可以为向量、矩阵、数据或数据框；`trim` 用于设置计算前去掉与均值差别较大数据的比例，默认为 0，表示计算中包括全部数据；`na.rm` 默认为 `FALSE`，表示不允许数据有缺失。

我们使用 2007.01–2012.09 时间段中国人寿股票价格的样本作为实例，具体说明用 R 如何做描述统计。这一数据集读入后包含两列，第一列为交易日期，第二列为当日收盘价：

```
> data=read.csv("d:/data/中国人寿股价.csv")
```

```

>attach(data)
The following object is masked from data (position 3):
Clsprc, Trddt
>price=Clsprc[!is.na(Clsprc)] #获得价格向量,同时去掉缺失数据
>mean(price)
[1] 27.83326

```

有时计算均值时,我们希望赋予各变量的权重是不一样的,也就是要计算加权平均数,这时用到的指令是 `weighted.mean(x,w,...)`, 其中 `x` 是样本数据, `w` 是与 `x` 长度相等的数值型向量,用于指出赋予样本各数据的权重值。

中位数通过函数 `median()` 计算,该函数与 `mean()` 的使用方法相同。由计算结果发现,均值略大于中位数,可以初步判断,中国人寿股价样本呈右偏分布。

```

>median(price)
[1] 24.02

```

分位数通过 `quantile()` 计算, `x` 是数值向量, `probs` 给出相应的百分位数,默认值是 `(0,0.25,0.5,0.75,1)`, `na.rm=FALSE` 表示不允许包含缺失数据。

默认条件下,可以直接计算出“五数”: 最小值、25%的四分位数、中位数、75%的四分位数和最大值。函数 `fivenum()` 用来计算五数。

```

quantile(x, probs = seq(0, 1, 0.25), na.rm = FALSE, names = TRUE, type = 7, ...)
>quantile(price)
 0%   25%   50%   75%   100%
14.740 19.285 24.020 31.160 75.080
>fivenum(price)
[1] 14.74 19.27 24.02 31.16 75.08
>min(price);max(price)
[1] 14.74
[1] 75.08

```

这样一个一个地输入函数非常不方便, R 提供了总体描述的函数。函数 `summary()` 可以计算出一组数据的五数和均值。

```

>summary(price)
Min.1st Qu. Median Mean3rd Qu.  Max.
 14.74   19.28   24.02   27.83   31.16   75.08

```

另外,在 R 中没有直接的函数用来计算众数。对于离散型样本数据,可以简单地手动计算,使用指令 `which.max(table(x))`。但像本例中这样的连续型样本,众数的定义是其分布的密度函数峰值对应的取值,这就需要我们自己写一个函数 (function) 来计算,然而出于描述数据分布形态的目的,我们大可以直接画出直方图来观察众数的大概位置。

5.3 离散趋势的分析

集中趋势只是数据分布的一个概括性度量，它反映的是各变量值向中心值聚集的程度。而变量的分散程度如何度量呢？分散程度，即各观测值远离中心值的程度，也称为“离中趋势”。离散程度越小，数据的代表性就越好。

5.3.1 离散趋势的测度

数据分布的离散程度主要靠极差、四分差、平均差、方差、标准差等统计指标来度量。在实际分析中，离散程度分析主要有以下作用：

- 衡量平均指标的代表性；
- 反映社会经济活动的均衡性；
- 研究总体标志值分布偏离正态的情况；
- 抽样推断等统计分析的一个基本指标。

（1）极差

首先从最简单的统计量入手，极差是描述数据离散程度最简单的测度值，是样本数据中两个极端值之差，也称为全距。数据越分散，极差越大。

$$R = x_{\max} - x_{\min}$$

极差计算简单又容易理解，但它只是利用了数据两端的信息，容易受极端值的影响，并且没有充分利用数列的信息。

（2）四分位差

数据的四分位差是两个四分位点之差，反映了中间 50% 数据的离散程度，其数值越小，说明中间的数据越集中。

$$Q_d = Q_l - Q_u, \quad Q_l \text{ 和 } Q_u \text{ 分别表示上下四分位数}$$

虽然四分位差避免了极端数据的影响，但“掐头去尾”后仍然丢失很多信息。

（3）方差与标准差

描述离散程度，最常用的指标是方差和标准差，它们利用了样本的全部信息去描述数据取值分散性。方差是各样本相对均值的偏差平方和的平均，即

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

方差的平方根，称为标准差。与方差不同的是，标准差是具有量纲的，它与变量值的计量单

位相同，因此具有较强的实际意义，在实际分析中会比较常用。

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$$

(4) 变异系数

一组数据的标准差与相应平均数之比，称为变异系数，也叫做离散系数。它是刻画数据相对分散性的一种度量，记为 CV。既然是“相对”，说明它去除了单位的影响，因此是个无量纲的统计量，用百分数表示。在实际应用中可以消除由于不同计量单位、不同平均水平所产生的影响。

$$CV = 100 \times \frac{s}{\bar{x}} (\%)$$

5.3.2 R 语言实现

可以通过函数 `range()` 计算极差。给出最小值和最大值两个点，再相减得到。

```
> m=range(price);m[2]-m[1]
[1] 60.34
>max(price)-min(price)
[1] 60.34
```

四分位差同样需要手动计算，比较便捷的方法是直接使用函数 `fivenum()`。

```
> q=fivenum(price);q[4]-q[2]
[1] 11.89
```

R 中的方差函数和标准差函数分别是 `var()` 和 `sd()`，非常简单。还可以手动输入公式，计算验证一下。

```
var(x, y = NULL, na.rm = FALSE, use)
sd(x, na.rm = FALSE)
>var(price);sd(price)
[1] 138.7829
[1] 11.78061
>sqrt((sum(price^2)-(sum(price))^2/length(price))/(length(price)-1))
[1] 11.78061
```

除了上面几个最常用的离散趋势统计量外，R 还有一个比较特殊的函数，即离差 `mad()`，它用于计算中位数绝对偏差，具有渐近正态的一致性。

```
mad(x, center = median(x), constant = 1.4826, na.rm = FALSE, low = FALSE, high = FALSE)
```

其中的参数 `center` 可选，默认为中位数，即计算样本偏离中位数的程度；`constant` 是比例因子，默认为一个常数；参数 `low/high` 默认取值为 `FALSE`，若是 `TRUE`，表示对于偶数个样本数据，中位数不取两个中间值的平均，而是采用较小/大的那一个。在默认状态下，R 中离差的计算公式

相当于

$$mad = 1.4826 \times median(abs(x - median(x)))$$

其中系数 1.4826 约等于 $1/\text{qnorm}(3/4)$ ，这样离差 `mad()` 在估计方差时才具有渐近正态的一致性。

```
>mad(price)
[1] 7.813302
```

5.4 数据的分布分析

5.4.1 分布情况的测度

仅通过集中趋势和离散趋势两个指标，仍无法判断数据的分布形态。数据比较集中的区域在分布图上就会形成一个“峰”，这个“峰”可能偏左，也可能偏右；峰值可能较高，也可能较低，这时就需要用到偏态（Skewness）和峰度（Kurtosis）两个统计方法对数据分布特征作进一步描述。

（1）偏度

前面提到，通过均值和平均数、众数之间的关系可以大致判断出数据点分布状态是对称、左偏还是右偏。显然，判断偏斜方向并不难，但要判断偏斜的程度却需要更精确的测度，即偏度系数。样本的偏度系数记为 SK ，计算公式为

$$SK = \frac{n}{(n-1)(n-2)s^3} \sum_{i=1}^n (x_i - \bar{x})^3 = \frac{n^2 \mu_3}{(n-1)(n-2)s^3}$$

其中， s 是标准差， μ_3 是样本的 3 阶中心矩。

具体的判断方法是：如果样本的偏度系数 $SK=0$ ，则说明数据关于均值对称；当 $SK<0$ 时，样本是左偏分布（Negative Skew）；当 $SK>0$ 时，表示正偏离程度较大，说明样本是右偏分布（Positive Skew），如图 5.2 所示。

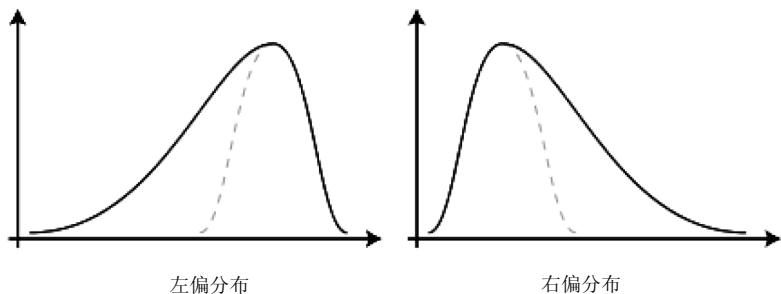


图 5.2 偏态分布

(2) 峰度

偏度系数判断尖峰在水平方向上往哪边偏移，可以说它是从横向的角度描述分布特点，那么从纵向的角度，描述数据分布的平坦或尖峰程度（称为峰态），就要通过峰度系数来衡量。记为 K 。

$$K = \frac{n^2(n+1)\mu_4}{(n-1)(n-2)(n-3)s^4} - 3 \frac{(n-1)^2}{(n-2)(n-3)}$$

其中， μ_4 是样本 4 阶中心矩， s^4 是样本标准差的四次方。对于分组数据，利用组中值 M_i 计算峰度系数， f_i 代表组内的频数。

$$K = \frac{\sum_{i=1}^k (M_i - \bar{x})^4 f_i}{ns^4} - 3$$

峰态通常是与标准正态分布相比较而言的，如果一组数据服从标准正态分布，则峰度系数 $K=0$ ；当分布比正态更加尖峰时， $K>0$ ；当分布比较平时， $K<0$ 。如图 5.3 所示，当 $K>0$ 时，分布为尖峰形状，因此分布两侧的尾部数据较少（Thin tails）；而 $K<0$ 时，分布的峰值较低，造成尾部较厚（Fat tails）。

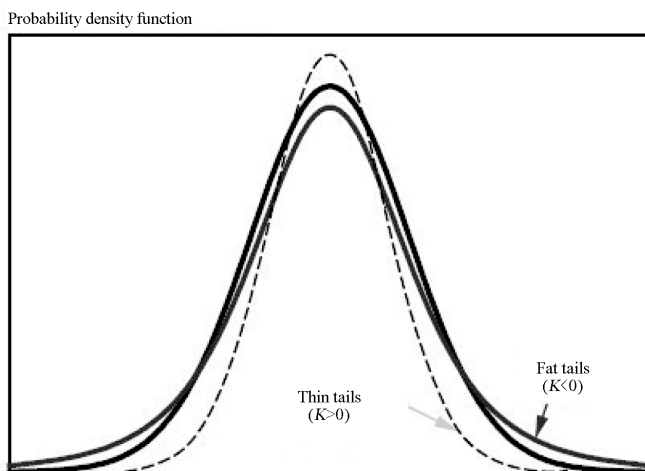


图 5.3 峰态分布

5.4.2 R 语言实现

在程序包 `timeDate` 中（或直接加载 `fBasics` 程序包），有直接计算偏度和峰度系数的函数，为 `skewness()` 和 `kurtosis()`。

也可以自己写一个函数 `function()` 进行验证，由于计算方法上的一些差别，两种方法计算的结

果有出入，但不影响对样本分布形态的判断。

```
>library(timeDate) #加载程序包
>skewness(price) #计算偏度系数
[1] 1.683351
attr(,"method")
[1] "moment"
> SK=function(x){
+   n=length(x);m=mean(x)
+   C=n/((n-1)*(n-2))*sum((x-m)^3)/(sd(x))^3;C
+ }
>SK(price) #手动计算偏度系数
[1] 1.686996

>kurtosis(price)
[1] 2.578657
attr(,"method")
[1] "excess"
> K=function(x){
+   n=length(x);m=mean(x);s=sd(x)
+   g2=((n*(n+1))/((n-1)*(n-2)*(n-3))*sum((x-m)^4)/s^4-(3*(n-1)^2)/((n-2)*(n-3)))
+   g2
+ }
>K(price)
[1] 2.600382
```

偏度系数 $SK=1.687>0$ ，说明样本是右偏分布，并且右偏程度比较严重。峰度系数 $K=2.60>0$ ，说明分布比正态更尖峰。

5.5 图形分析及 R 实现

电影《达芬奇密码》中有这样一句话：“A picture says more than a thousand words!”意思是，利用图形描述数据规律一目了然，胜过千言万语。图形显示的结果可以直观地概括上述指标所反映的分布特点，对于一组数据的描述，主要通过直方图、茎叶图等。

5.5.1 直方图和密度函数图

获得了一组样本数据，我们最直接的想法是通过频率分布直方图观察其样本的分布，常与直方图配套出现的是核密度估计函数 `density()`，它可以根据已知样本，去估计样本的密度函数曲线。结合直方图和密度估计曲线，可以得到和统计指标所反映的相似结论：样本呈现严重的右偏分布，右尾拖得很长；数据有一较高的峰值。

```
>hist(price,breaks=50,prob=T) #参数 breaks 设置直方图的组距，prob=T 规定绘制密度直方图
>lines(density(price),col="blue") #利用核密度估计函数 density()，绘制密度曲线图
```

绘制结果如图 5.4 所示。

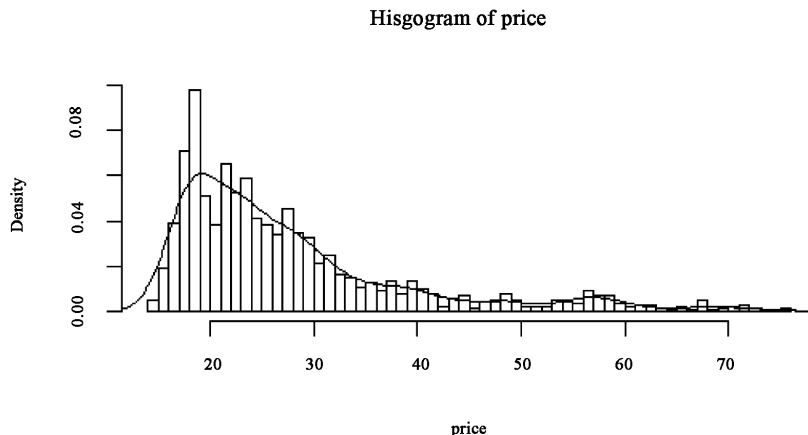


图 5.4 直方图和密度函数曲线

5.5.2 QQ 图

QQ 图用于直观验证一组数据是否来自某个分布，或者验证某两组数据是否来自同一族的分布。在教学和软件中常用 QQ 散点图来检验数据是否来自于正态分布。假定总体为正态分布 $N(\mu, \sigma^2)$ ，对于样本 x_1, x_2, \dots, x_n ，其顺序统计量为 $x_{(1)}, x_{(2)}, \dots, x_{(n)}$ 。 $\Phi(x)$ 是标准正态分布 $N(0,1)$ 的分布函数，而 $\Phi^{-1}(x)$ 是其反函数，QQ 图即是由以下的点构成的散点图：

$$\left(\Phi^{-1}\left(\frac{i-0.375}{n+0.25}\right), x_{(i)} \right), \quad i = 1, 2, \dots, n$$

QQ 图是正态分位数-分位数图，横轴是理论值，纵轴是样本值，若样本数据近似服从正态分布，那么 QQ 图上的散点应均匀地分布在直线 $y = \sigma x + \mu$ 附近，这条直线的斜率是正态分布的标准差 σ ，截距是均值 μ 。

R 软件中的函数 `qqnorm()` 和 `qqline()` 可以分别用于绘制正态 QQ 散点图和相应直线，其使用方法为：

```
qqnorm(y, ylim, main="Normal Q-Q Plot", xlab="Theoretical Quantiles",
       ylab="Sample Quantiles", plot.it = TRUE, datax = FALSE, ...)
qqline(y, datax = FALSE, distribution = qnorm, probs = c(0.25, 0.75), qtype = 7, ...)
qqplot(x, y, plot.it = TRUE, xlab = deparse(substitute(x)), ylab =
       deparse(substitute(y)), ...)
```

其中 `x, y` 为数据样本；`xlab, ylab` 和 `main` 是图标签，分别用于设置 X 轴和 Y 轴的标签、图像标题。其它参数用法参见第四章 `plot()` 的参数设置。

对股票价格样本绘制 qq 散点图和曲线图, 判断它是否来自于正态总体。

```
> qqnorm(price)
> qqline(price)
```

绘制结果如图 5.5 所示。

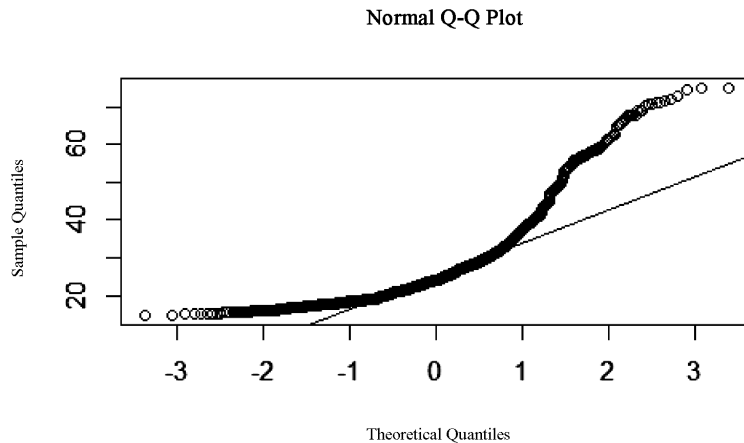


图 5.5 股票价格的正态 QQ 图

从图 5.5 可以看出, QQ 散点图并不均匀地分布在直线两侧, 因此股价样本不服从正态分布。接下来举一个正态分布样本的示例。

```
> a=c(21.0,21.4,22.5,21.6,19.6,20.6,20.3,19.2,20.2,21.3)
> qqnorm(a);qqline(a)
```

绘制结果如图 5.6 所示。

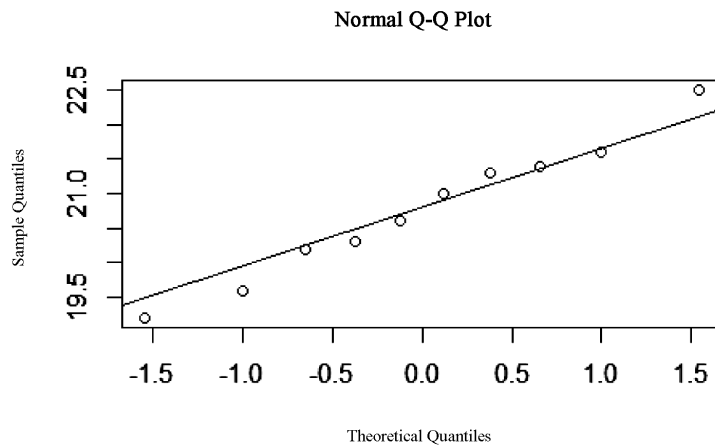


图 5.6 正态 QQ 图

从图 5.6 可以看出，样本数据是来自于正态总体。

5.5.3 茎叶图

茎叶图也是考察样本分布的重要方法，由统计学家约翰·托奇设计。它是将数组中的数按位数进行比较，将数的大小基本不变或变化不大的位作为一个“茎”，将位变化大的数作为“叶”，列在茎的后面，这样就可以清楚地看到每个茎后面有多少样本数，以及具体数值是多少。

茎叶图更适用于描述样本量较小的分布，在本例中，股票价格样本有逾千个，茎叶图不能完全显示，所以抽取其中的 50 个样本作图，R 中用函数 `stem()` 绘制茎叶图。

```
stem(x, scale = 1, width = 80, atom = 1e-08)
```

其中，`x` 是数据向量，`scale` 控制茎叶图的长度，`width` 控制绘图的宽度，`atom` 是容差。

```
>set.seed(111) #设置抽样种子
>s=sample(price,50) #从数据集 price 中抽取 50 个样本
> stem(s)
```

结果如下所示：

```
The decimal point is 1 digit(s) to the right of the |
1 |
1 | 5888999999999
2 | 01112223334444
2 | 555556788889
3 | 022
3 | 788
4 | 1
4 | 8
5 | 3
5 | 69
```

在上面的结果中，左边的“茎”是股价的十位数字，右边的“叶”是个位数字。茎叶图从外观上来讲，比较像是横放的直方图，我们所抽取的样本也是一个右偏分布。“叶”的每一个数字都代表一个样本，显示频数的个数，所以从上向下数第 25、26 个数字的平均可以体现样本中位数的值。

5.5.4 箱线图

箱线图在描述统计中也是非常重要的图形，可以说它的重要程度不亚于直方图。箱线图是“五数”的一个图形概括，可以用来对数据分布的形状进行大致判断，是直观展现数据分布主要特征的方式。在上一章已经介绍过，R 中使用函数 `boxplot()` 绘制箱线图。

函数 `boxplot()` 有三种使用方法，第一种最为直观和简单，因此也最常用：

```
boxplot(x, ...)
```

其中, x 是数值型向量、列表或数据框。第二种形式为:

```
boxplot(formula, data = NULL, ..., subset, na.action = NULL)
```

formula 是公式, 形如 $y \sim \text{grp}$, 这里 y 是数值型向量, 而 **grp** 通常为因子, 用来表示数据的分组, 这种形式的绘图结果是分类的箱线图, 便于比较。第三种形式比较复杂:

```
boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE, notch = FALSE, outline = TRUE,
names, plot = TRUE, border = par("fg"), col = NULL, log = "",
pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),
horizontal = FALSE, add = FALSE, at = NULL)
```

其中, x 是数值型变量、列表或数据框; **range** 是“触须”的范围, 默认值为 1.5; **notch** 是逻辑变量, 默认值 **FALSE** 表示绘制的箱线图不带有切口; **outline** 也是逻辑变量, 默认值为 **TRUE**, 即标明异常值点; **col** 用于设置不同颜色 (根据不同的数值); **horizontal** 是逻辑变量, 默认值为 **FALSE**, 设置成 **TRUE** 时将箱线图绘制成水平状。

要绘制出本章股票价格样本的箱线图, 在 R 中输入如下指令:

```
> boxplot(price, main = "Boxplot of price")
```

绘制结果如图 5.7 所示。

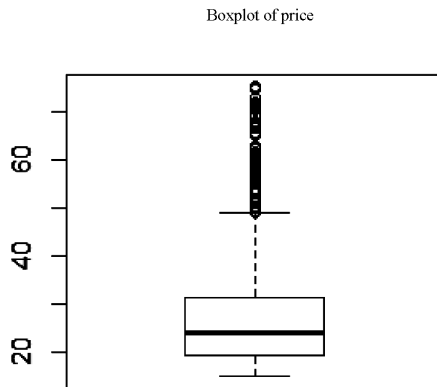


图 5.7 股票价格的箱线图

在图 5.7 的箱线图中, 75%和 25%四分位数分别为中间箱体的上下两条线, 箱体中间的粗线段是中位数 M_e 所在的位置。箱体外的两条线是数据的最大值和最小值, 但不得超过 1.5 倍四分位数间距, 若是超出了则标记为异常值点, 用“o”表示。

在本例中, 股价分布的右尾很长, 因此就出现了较多的异常值点, 这是在实际数据分析中很有可能出现的情况。

5.5.5 经验分布图

直方图主要适用于总体为连续分布的样本。对于更一般的总体分布，如果要估计其分布函数，可以使用经验分布函数（edf）。在 R 中函数 `ecdf()` 给出样本的经验分布，通过 `plot()` 绘制。

```
ecdf(x)
plot(x, ..., ylab="Fn(x)", verticals = FALSE, col.0line = "gray70", pch = 19)
```

通过 `ecdf()` 生成经验分布函数 $F_n(x)$ 在各样本点的值，是一个数值型向量；`plot()` 中的 `verticals` 是逻辑变量，若为 `TRUE` 表示画出竖线，默认是不画竖线。

```
> plot(ecdf(price), main="empirical cdf", xlab="price", ylab="Fn(price)")
> x=min(price):max(price)
> lines(x, pnorm(x, mean(price), sd(price)), col="red") #正态分布函数曲线
> legend(60, 0.4, legend=c("样本分布", "正态分布"), lty=1, col=1:2)
```

绘制结果如图 5.8 所示。

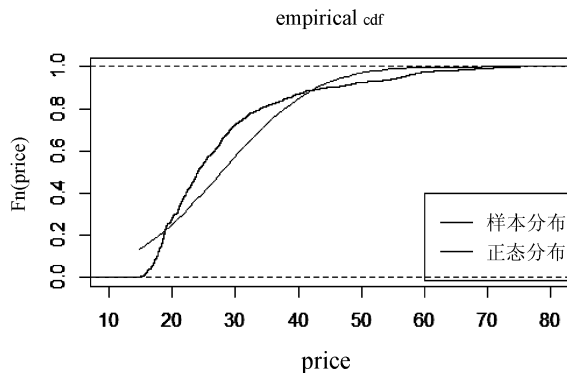


图 5.8 经验分布图

5.6 多组数据分析及 R 实现

5.6.1 多组数据的统计分析

当数据分为多个组别时，其描述统计与单组数据有相通之处，但在体现变量关系方面还是有一定差别的。上文中使用的例子是一组数据，假设按照年份将之分组，重新读入 2009、2010 和 2011 这三年的完整数据，在前面的理论基础上，对三组数据进行描述性统计分析。由于每年的交易日不完全相同，为简单处理，把有缺失值的样本删去。

```
> group=read.csv("d:/data/09-11 股价.csv")
> group=na.omit(group) #忽略缺失样本
```

首先，与单组情形类似，直接使用 `summary()` 可以得到各组数据的均值和“五数”。通过观察

横向数据，我们可以对比不同年份股票价格的基本统计量。以均值为例，它反映出股票价格的平均水平是逐年下降的。

```
>summary(group)
      Price09      Price10      Price11
Min. :18.67  Min.      :20.90  Min.      :14.74
1st Qu.:23.08  1st Qu.      :23.25  1st Qu.      :16.98
Median :27.45  Median      :24.70  Median      :18.53
Mean   :26.51  Mean        :25.20  Mean        :18.72
3rd Qu.:29.96  3rd Qu.      :27.40  3rd Qu.      :21.05
Max.   :34.01  Max.        :31.42  Max.        :22.33
```

函数 `var()` 应用在多组数据上，得到的计算结果是一个协方差阵，其每个元素是各个向量之间的协方差。使用指令 `cor(group)` 也得到相同结果。

```
> options(digits=3) #设置显示格式：数字只显示3位
>var(group)
      Price09 Price10 Price11
Price09 16.55 -7.18 -6.75
Price10 -7.18  5.99  4.16
Price11 -6.75  4.16  4.84
```

协方差的大小在一定程度上反映了变量之间的相互关系，但它还受变量本身度量单位的影响，因此我们还要计算相关系数来度量变量之间的线性相关程度。在 R 中使用函数 `cor()` 计算相关系数矩阵。

```
cor(x, y = NULL, use = "everything", method = c("pearson", "kendall", "spearman"))
```

其中，`x,y` 是计算的对象，当 `x` 是一个数据框或列表时 `y` 可以省略；`use` 指定如何处理缺失样本；`method` 给出计算哪一种相关系数：默认的皮尔逊（Pearson）系数度量线性相关性，如果数据呈现的不是线性关系，而是单调的，则可以用肯德尔（Kendall）或斯皮尔曼（Spearman）相关系数，它们描述的是秩相关性。

本例中没有理论基础可以说明两年的股价会呈线性关系，所以计算 Spearman 系数。

```
>cor(group,method="spearman")
      Price09 Price10 Price11
Price09 1.000 -0.687 -0.710
Price10 -0.687 1.000  0.752
Price11 -0.710 0.752  1.000
```

在上面的矩阵中，对角线都是 1，因为变量与自身是完全相关的。而 2009 年股价与 2010 年股价的相关系数是 -0.687，说明这两年的股票价格呈负相关的关系。

5.6.2 多组数据的图形分析

常用的图形表示方法大多用于一维数据的描述，但实际问题中的数据集往往是多变量样本。

接下来的这一节，将介绍在实践中常用的多组数据绘图。

(1) 二维散点图

首先，两组数据的关系用散点图可以清楚地描述。在散点图中加入一条拟合曲线有助于更好地把握变量关系。

R 中的函数 `lowess()` 可通过加权多项式回归对散点图进行平滑，拟合一条非线性的曲线，但其只能适用于二维情况。与之类似的 `loess()` 用于处理多维情况。

```
lowess(x, y = NULL, f = 2/3, iter = 3, delta = 0.01 * diff(range(x)))
```

`x, y` 指定两个向量；`f` 是平滑的跨度，值越大，曲线的平滑程度越高；`iter` 控制应执行的迭代数，值越高平滑越精确，但使用较小的值会使程序跑得比较快。

```
> attach(group)。
> plot(Price10~Price09,xlab="Price of 2009",ylab="Price of 2010")
> lines(lowess(Price09,Price10),col="red",lwd=2) #拟合曲线
```

绘制结果如图 5.9 所示。

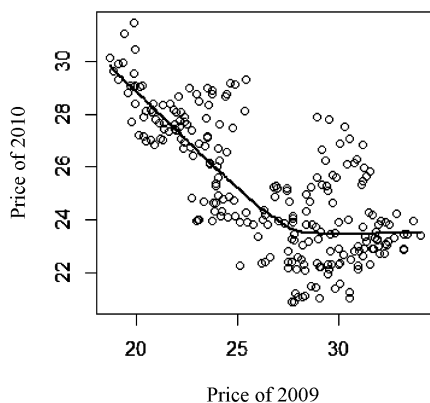


图 5.9 两变量散点图和平滑曲线

从散点图中可以看出，2009 年和 2010 年的股票价格并不是完全线性的关系，这与由 Spearman 系数得出的结论相同，两年的股价具有负的相关性，价格变动趋势相反。

(2) 等高线图

有时候数据量很大，散点图上的数据点就会非常集中，不容易看出变量的关系或趋势，这就需要借助二维等高线图来描述。首先利用程序包 MASS 中的函数 `kde2d()` 来估计出二维数据的密度函数，再利用函数 `contour()` 画出密度的等高线图。如果不想画出图上的数据标签，可以将参数 `drawlabels=FALSE` 去掉。函数 `kde2d()` 的使用方法如下：

```
kde2d(x, y, h, n = 25, lims = c(range(x), range(y)))
```

其中 x, y 分别为横轴和纵轴的数据； n 指定每个方向上的网格点数量，可以是标量或长度为 2 的一个正数向量；参数 $lims$ 表示横纵轴的范围。

接下来就利用上述的两个函数绘制 2009 年和 2010 年股价的二维等高线图。在 R 中输入指令：

```
>library(MASS)
>a=kde2d(Price09,Price10)
>contour(a,col="blue",main="Contour plot",xlab="Price of 2009",ylab="Price of 2010")
```

绘制结果如图 5.10 所示。

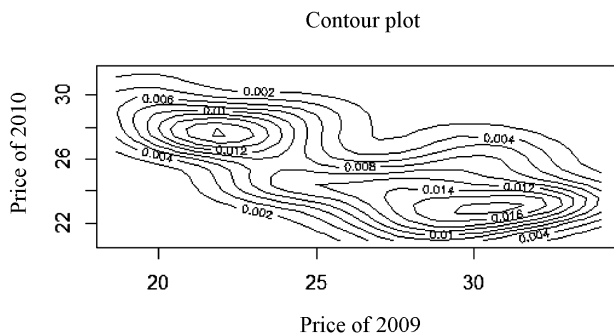


图 5.10 两变量等高线图

与地理上的等高线地形图相同，即等高线越靠近内部说明密度越高，散点分布越集中。

(3) 矩阵散点图

以上两种图形主要描述二维数据，实际中的数据集往往是多维的，我们使用的例子也是一样，多组数据的图形也可以用散点图来展示，不同之处在于这里是矩阵散点图。对于一个数据框，R 中可以直接使用 `plot()` 命令或 `pairs()` 绘制矩阵散点图。

在 R 中输入如下指令来绘制 2009—2011 年多组股价数据的矩阵散点图：

```
>plot(group,main="Scatterplot Matrices")
```

或

```
>pairs(group)
```

绘制结果如图 5.11 所示。

图 5.9 绘制出了三个年份之间股价的矩阵散点图，其中上三角与下三角的三幅图是对称的，分别为 2009—2010、2010—2011、2009—2011 年中国人寿股价的二维散点图。矩阵散点图的特点主要是直观简便，把多个二维散点图以矩阵的形式排列，也有助于比较两两变量之间的相关关系。

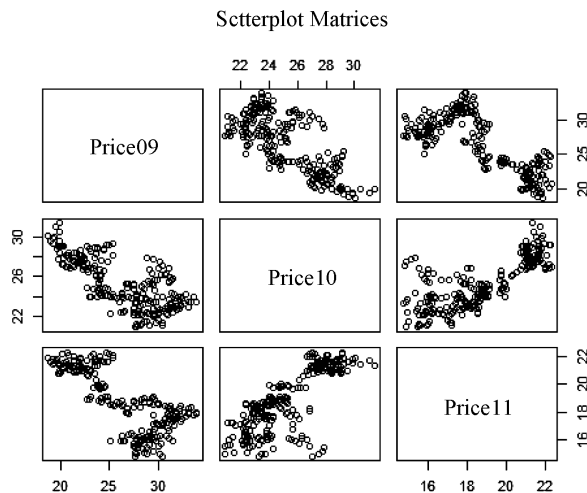


图 5.11 多组数据的矩阵散点图

(4) 矩阵图

在处理多组数据时，常将各组数据放在一起进行比较，`matplot()`可将各变量的散点图放在同一个绘图区域中。

上文计算的相关系数矩阵反映出 2009 年和 2010 年的股价成负相关，而 2010 年 2011 年的股价则变动趋势相同，这些结论从图 5.10 中也可以得到，黑色曲线表示 2009 年股价呈上升趋势，而 2010 和 2011 年股票价格则整体下跌，在 2011 年更是创出了历史新低。

```
> matplot(group, type="l", main="Matplot") #type="l" 表示绘制曲线而不是散点
> legend(0, 35, legend=2009:2011, pch="—", cex=0.6, col=1:3) #cex 指定字体大小
```

绘制结果如图 5.12 所示。

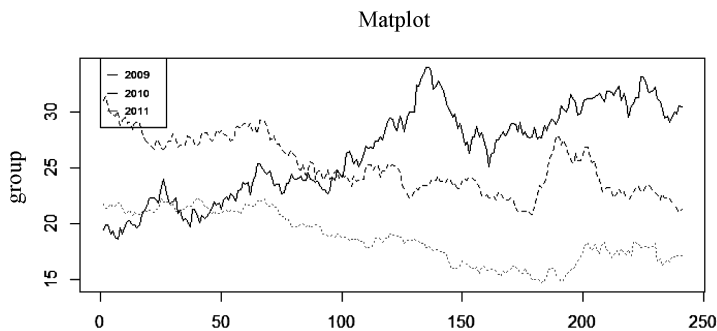


图 5.12 多组数据绘图

(5) 箱线图

在 5.5.4 节中已经详细介绍过箱线图, 函数 `boxplot()` 也可以用于多组数据的绘图, 在同一区域中画出各变量的箱线图, 便于对比。箱线图的好处在于可以同时对比最值、均值、中位数和整体分布情况, 尤其适用于时间序列的对比, 从图 5.13 中的几条“线”可以看出 2011 年的股票价格从最大值、75%分位点到中位数、最小值等都低于 2009 年和 2010 年, 因此 2011 年的股价水平整体低于前两年。

```
>boxplot(group,cex.axis=0.6)
```

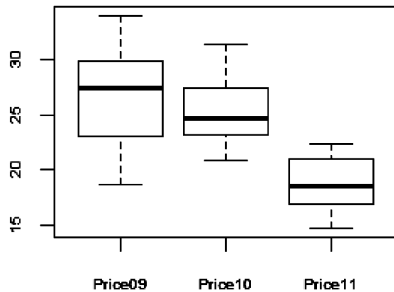


图 5.13 多组数据箱线图

(6) 星图 (雷达图)

假设变量是 p 维数据, 有 n 个观测值, 故第 k 次的观测值写作向量形式为

$$X_k = (x_{k1}, x_{k2}, \dots, x_{kp}), \quad k = 1, 2, \dots, n$$

星图作为一种多元数据的图示方法, 可以对 p 维数据进行比较。由于星图 (如图 5.14 所示) 看起来很像雷达图像, 因此也称为雷达图; 星图也像一个蜘蛛网, 所以有时也称为蜘蛛图。其绘制原理是:

- (1) 作一个圆, 并将圆周 p 等分。
- (2) 连接圆心和圆周上的 p 个分点, 把这 p 条半径依次视为变量坐标轴, 并标上适当的刻度。
- (3) 对给定的每一次观测值, 在上述 p 条坐标轴上分别标出 p 个变量的取值, 然后将它们连接成一个 p 边形, 即形成了一个“星”。
- (4) 对 n 次观测值都重复 (3) 的步骤, 就形成了 n 个星图。

R 中的函数 `stars()` 可以绘制星图, 其使用方法是:

```
stars(x, full = TRUE, scale = TRUE, radius = TRUE, labels = dimnames(x)[[1]],
      locations = NULL, nrow = NULL, ncol = NULL, len = 1,
      key.loc = NULL, key.labels = dimnames(x)[[2]], key.xpd = TRUE,
      xlim = NULL, ylim = NULL, flip.labels = NULL, draw.segments = FALSE,
```

```
col.segments = 1:n.seg, col.stars = NA, col.lines = NA, axes = FALSE,
frame.plot = axes, main = NULL, sub = NULL, xlab = "", ylab = "",
cex = 0.8, lwd = 0.25, lty = par("lty"), xpd = FALSE,
mar = pmin(par("mar"), 1.1+ c(2*axes+ (xlab != ""), 2*axes+ (ylab != ""), 1, 0)),
add = FALSE, plot = TRUE, ...)
```

`stars()`函数中包括很多的参数设置，这里着重介绍其中比较重要的参数，其他请参考 `help` 文档。其中 `x` 是数据矩阵或数据框；`full` 是逻辑变量，默认值为 `TRUE` 表示星图绘制成整圆的，`FALSE` 表示星图绘制成上半圆的图形；`scale` 也是逻辑变量，默认值 `TRUE` 表示数据矩阵的每一列是独立的，且每列的最大值为 1，最小值为 0，若值为 `FALSE` 则所有的星图会叠在一起；`radius` 是逻辑变量，默认值为 `TRUE`，表示画出构成星图半径的连线，`FALSE` 则画出的星图没有半径构成的连线；`len` 是半径刻度因子，设置星图的比例，默认值为 1；`key.loc` 是由 `x` 和 `y` 坐标值构成的向量（默认值为 1），设置星的位置；`draw.segments` 同样是逻辑变量，默认值为 `FALSE`，即用直线连接 p 个半径上的点，若为 `TRUE` 则画出的星图是一段一段的弧。

但是星图比较适用于维度较低的数据，方便比较。例如学校随机抽取 6 名学生的 5 门考试成绩，如表 5.2 所示，从 D 盘读入该数据集，我们将画出这 6 名学生成绩的星图。

表 5.2 学生考试成绩

科目 1	科目 2	科目 3	科目 4	科目 5
89	90	92	95	99
90	77	74	86	89
79	88	94	98	97
100	90	85	100	81
83	87	70	77	73
100	94	97	90	89

在 R 软件中输入如下指令：

```
> score=read.table("d:/data/score.txt",header=T) #读入数据
> stars(score) #绘制星图，所有参数均设置为默认值
```

绘制结果如图 5.14 所示。

星图的半径反映数值的大小，因此从图 5.14 中可以看出学生编号 1 和 6 的成绩较好，而第 5 名学生成绩最差，第 2 名其次。

设置函数 `stars()` 中的参数，可以将上面的星图绘制成图 5.15 所示的另一种形式。在 R 中输入指令：

```
> stars(score,full=FALSE,draw.segments=TRUE,key.loc=c(5,0.5),mar=c(2,0,0,0))
#full=F
# 表示绘制成半圆的图形，draw.segments=T 表示画出弧线
```

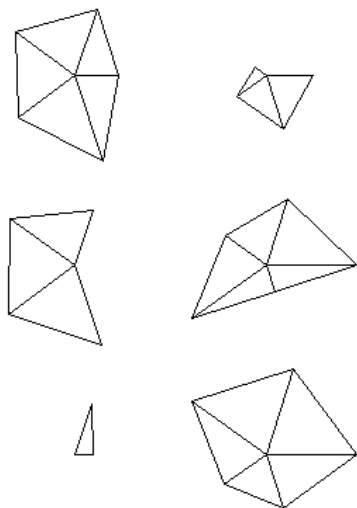


图 5.14 学生成绩的星图（形式 1）

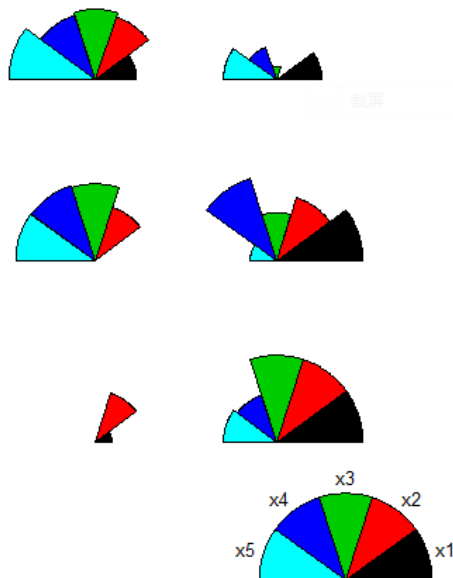


图 5.15 学生成绩的星图（形式 2）

（7）折线图

和星图相比，折线图是一种更加直观地展示多维图形的方法，但同样地，折线图比较适用于数据量较小的数据样本。

折线图由以下步骤绘制而成：

- （1）在直角坐标系的横轴上取 p 个点，表示 p 个变量；
- （2）对每一次观测值， p 个点的纵坐标即表示变量的取值；
- （3）连接 p 个点可以得到一条折线，即为该次观测值的折线图；
- （4）对于 n 次观测值，均重复上述步骤，即可得到 n 次观测值的折线图。

在 R 中并没有现成的绘制折线图的函数，因此需要我们自行编写，这里命名为函数 `outline()`。

```
outline=function(x){
  if(is.data.frame(x)==TRUE)
    x=as.matrix(x) #若 x 为数据框，则先转换为矩阵形式
  m=nrow(x);n=ncol(x) #提取 x 的行列数
  plot(c(1,n),c(min(x),max(x)),type="n",
  main="The outline graph of data",xlab="Number",ylab="Value")
  for(i in 1:m){
    lines(x[i,],col=i)
  }
}
```

其中的参数 x 是数据矩阵或数据框, 函数的运行结果是 n 次观测值的折线图。例如对上面学生成绩的例子绘制折线图, 则应在 R 中输入如下指令:

```
> outline(score)
```

绘制结果如图 5.16 所示。

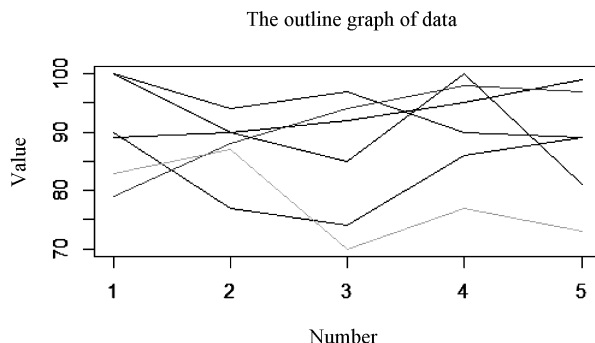


图 5.16 学生成绩的折线图

从图 5.16 中可以看出, 越是靠近图形上方的学生成绩越好。本节仅给出一个编写函数的示例, 若样本量较多, 编写函数时应在绘图中加入样本编号。这种折线图可以用于聚类分析的初步判断。

(8) 调和曲线图

调和曲线图是 D.F.Andrews 在 1972 年提出的三角多项式作图方法, 所以又称为三角多项式图, 其思想是把高维空间中的一个样本点对应于二维平面上的一条曲线。

设 p 维数据 $x = (x_1, x_2, \dots, x_p)'$, 对应的曲线是

$$f_x(t) = \frac{x_1}{\sqrt{2}} + x_2 \sin(t) + x_3 \cos(t) + x_4 \sin(2t) + x_5 \cos(2t) + \dots \quad (-\pi \leq t \leq \pi)$$

上式中, 当 t 在区间 $[-\pi, \pi]$ 上变化时, 其轨迹是一条曲线。

在多项式的图表示中, 当各变量的数值太悬殊时, 最好先标准化后再作图。这种图对聚类分析帮助很大, 如果选择聚类统计量为距离的话, 则同类的曲线非常靠近拧在一起, 不同类的曲线相互分开, 非常直观。

调和曲线图有两点优良数学性质: 一是保持线性关系, 二是与一般的欧式距离之间的关系。第一点从上面的公式中就可以直观得到结论, 第二个性质可通过计算得到。按照一般常用的范式, 定义样本 x 和 y 之间的距离为如下的平方积分形式

$$d_{f_x f_y}^2 = \int_{-\pi}^{\pi} |f_x(t) - f_y(t)|^2 dt$$

根据三角函数积分，通过运算最终可以得到这一距离与欧氏距离之间具有如下关系：

$$d_{f_x f_y}^2 = \pi d_{xy}^2$$

按照上式的描述，我们可以在 R 中自己编写调和曲线图函数 `unison()`。

```
unison=function(x){
  if (is.data.frame(x)==TRUE)
    x=as.matrix(x) #若 x 为数据框，则先转换为矩阵形式
  t=seq(-pi, pi, pi/30) #设置 t 的变化范围
  m=nrow(x); n=ncol(x) #提取 x 的行列数
  f=array(0, c(m,length(t))) #f 赋值为一个数组
  for(i in 1:m){
    f[i,]=x[i,1]/sqrt(2)
    for( j in 2:n){
      if (j%2==0)
        f[i,]=f[i,]+x[i,j]*sin(j/2*t)
      else
        f[i,]=f[i,]+x[i,j]*cos(j/2*t)
    }
  }
  plot(c(-pi,pi),c(min(f),max(f)),type="n",main ="The Unison graph of
Data",xlab="t",ylab="f(t)")
  for(i in 1:m) lines(t,f[i,],col=i)
}
```

其中，`x` 是数据矩阵或数据框。函数的运行结果是调和曲线图（如图 5.17 所示）。

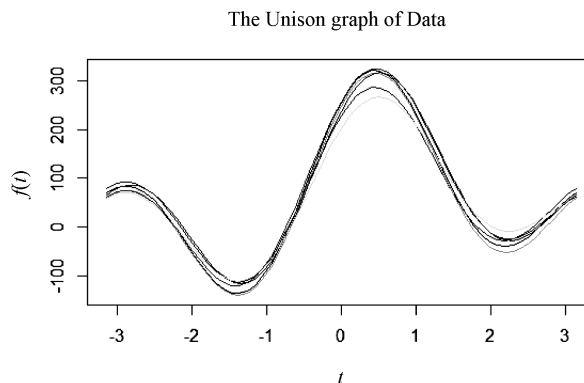


图 5.17 学生成绩的调和曲线图

图 5.17 虽然不能用于判断学生成绩的好坏，但这种调和曲线图对聚类分析有很大帮助，可以用于初步判断，即同类的曲线会聚集在一起，而不同类的曲线则会拧成不同的曲线束，比较直观。

第 6 章

参数估计及 R 实现

上一章介绍了数据的描述性分析。当我们对样本的分布有了大致了解后，下一步就是要根据样本去推断总体的分布和特征，这个过程称为统计推断。统计推断是与数据分析相关的主要推理形式。统计推断分两步进行：参数估计和假设检验。

参数估计是对所要研究的总体参数，进行合乎数理逻辑的推断。在很多实际分析中，我们取得样本数据后，对其总体的分布类型是已知的，但具体的分布完全由所含参数决定，此时写不出确切的概率密度函数，这就需要对未知的参数作估计。

本章将分别从点估计和正态总体的区间估计，介绍如何运用 R 中的函数完成运算。

6.1 点估计及 R 实现

参数估计有两类：一类是点估计，另一类是区间估计。例如，一批产品的废品率为 θ ，为了估计这一参数，从这批产品中随机地抽出 n 个样本作检查，以 X 表示其中的废品个数，那么 X/n 就是 θ 的一个点估计。

设 X_1, X_2, \dots, X_n 为总体 X 的样本，总体的分布函数 $F(x; \theta)$ 形式已知，其中 θ 为待估参数， x_1, x_2, \dots, x_n 为对应的样本观测值。接下来的问题就是构造一个适当的统计量，用样本观测值来估计参数 θ 的取值。这种对未知参数的定点估计称为未知参数的点估计。

6.1.1 矩估计

根据辛钦大数定律和强大数定理，如果总体 X 的 k 阶矩存在，那么样本的 k 阶矩依概率收敛于总体 k 阶矩，样本矩的连续函数收敛于总体矩的连续函数，这为矩估计方法提供了理论支持。英国统计学家 K. Pearson 在 20 世纪初提出，以相应的样本矩作为总体各阶原点矩的估计量，即矩

估计。

首先，做参数估计时需要注意原点矩的存在性，只有保证分布的矩存在我们才能继续矩估计方法。设总体 X 分布含有 m 个未知参数 $\theta = (\theta_1, \theta_2, \dots, \theta_m)$ ，根据分布函数 $F(x; \theta)$ 可以计算出总体的 k 阶矩，它们都是 $\theta_1, \theta_2, \dots, \theta_m$ 的函数，即

$$E(X^k) = f(\theta_1, \theta_2, \dots, \theta_m), \quad k = 1, 2, \dots, m$$

而样本的 k 阶矩是

$$A_k = \frac{1}{n} \sum_{i=1}^n X_i^k, \quad k = 1, 2, \dots, m$$

用 A_k 去估计 $E(X^k)$ ，则有

$$f(\theta_1, \theta_2, \dots, \theta_m) = E(X^k) = A_k = \frac{1}{n} \sum_{i=1}^n X_i^k, \quad k = 1, 2, \dots, m$$

解方程组，从而可以得到关于未知参数的解

$$\hat{\theta}_i = \hat{\theta}_i(X_1, X_2, \dots, X_n), \quad i = 1, 2, \dots, m$$

这样得到的 $\hat{\theta} = (\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_m)$ 就是总体真实参数的矩估计值。

由于不同分布造成函数 f 形态各异，因此在 R 中不可能有固定的 R 程序来直接得到矩估计结果，只能利用 R 的计算功能具体问题具体分析。这里首先介绍几个 R 中常用的解方程函数，如表 6.1 所示。

表 6.1 R 中的解方程函数

函 数	功 能	程序包
uniroot()	求解一元（非线性）方程	stats
multiroot()	给定 n 个（非线性）方程，求解 n 个根	rootSolve
uniroot.all()	在一个区间内求解一个方程的多个根	rootSolve
BBsolve()	使用 Barzilai-Borwein 步长求解非线性方程组	BB

其中 `BBsolve()` 与给定的参数初值有关，在实际中并不常用，因此这里重点介绍前两个函数的使用方法。首先是解一次方程的函数 `uniroot()`：

```
uniroot(f, interval, ..., lower = min(interval), upper = max(interval),
       f.lower = f(lower, ...), f.upper = f(upper, ...),
       tol = .Machine$double.eps^0.25, maxiter = 1000)
```

其中 f 指定所要求解方程的函数； $interval$ 是一个数值向量，指定要求解的根的范围；或

者用 `lower` 和 `upper` 分别指定区间的两个端点；`tol` 表示所需的精度（收敛容忍度）；`maxiter` 为最大迭代次数。

如果遇到多元方程的求解，就需要利用 `rootSolve` 包的函数 `multiroot()` 来解方程组。`multiroot()` 用于对 n 个非线性方程求解 n 个根，其要求完整的雅可比矩阵，采用 Newton-Raphson 方法。其调用格式为：

```
multiroot(f, start, maxiter = 100, rtol = 1e-6, atol = 1e-8, ctol = 1e-8, useFortran = TRUE,...)
```

`f` 指定所要求解的函数；由于使用的是牛顿迭代法，因而必须通过 `start` 给定根的初始值，其中的 `name` 属性还可以标记输出变量的名称；`maxiter` 是允许的最大迭代次数；`rtol` 和 `atol` 分别为相对误差和绝对误差，一般保持默认值即可；`ctol` 也是一个用于控制迭代次数的标量，如果两次迭代的最大变化值小于 `ctol`，那么迭代停止，得到方程组的根。

```
xx
```

接下来用实例来说明在 R 中求矩估计的方法，由于样本分布的多样性，矩估计必须“具体问题具体分析”，并没有一个固定的求解套路。

例如，已知某种保险产品在一个保单年度内的损失情况如表 6.2 所示，其中给出了不同损失次数下的保单数，我们对损失次数的分布进行估计。已知分布类型是泊松（Poisson），其样本均值即为参数 λ 的矩估计。

$$E(X) = \lambda = \bar{X}$$

表 6.2 损失次数数据

损失次数	0	1	2	3	4	5
保单数	1532	581	179	41	10	4

```
>num=c(rep(0:5,c(1532,581,179,41,10,4))) #用 rep() 函数生成样本，样本值有 0~5 的数字构成，
函数中的第二个向量对应表示每个数字的重复次数
>lambda=mean(num)
>lambda
[1] 0.4780571
```

根据估计的参数值，我们可以画图比较损失次数的估计值和样本值之间的差别。

```
> k=0:5
>ppois=dpois(k,lambda)
>poisnum=ppois*length(num) #由 poisson 分布生成的损失次数
> plot(k,poisnum,ylim=c(0,1600)) #画图比较，为图形效果更好，用参数 ylim 设置纵轴的范围，最小值为 0，最大值要大于样本的最值，选取 1600
>samplenum=as.vector(table(num)) #样本的损失次数
>points(k,samplenum,type="p",col=2)
>legend(4,1000,legend=c("num","poisson"),col=1:2,pch="o")
```

绘制结果如图 6.1 所示。

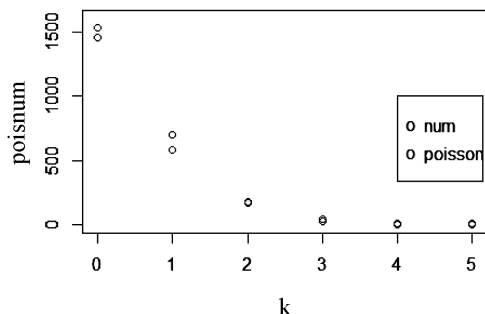


图 6.1 损失次数的矩估计

泊松分布只包含一个参数值，所以只需要一阶矩（即均值）就可以完成估计。参数的个数与需要的阶数是对应的，例如随机变量 X 服从 $[\theta_1, \theta_2]$ 的均匀分布，现有 n 个样本 $x = (x_1, \dots, x_n)$ ，为了估计 θ_1, θ_2 ，利用分布的一阶原点矩估计均值，加上二阶中心矩估计方差

$$E(X) = \frac{\theta_1 + \theta_2}{2} = \bar{x}$$

$$Var(X) = \frac{(\theta_2 - \theta_1)^2}{12} = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 = S^2$$

涉及多个参数的矩估计问题，解方程组是最大的难点，这时我们需要开发 R 的计算功能，前面提到 rootSolve 包的函数 multiroot() 用于解方程组。在 R 中使用它时我们要注意，multiroot() 求解的是函数值为 0 时方程组的根，所以编写函数 function 时要注意减去样本均值/方差。

下面以均匀分布样本 x 为例具体说明。

```
>install.packages("rootSolve")
> x=c(4,5,4,3,9,9,5,7,9,8,0,3,8,0,8,7,2,1,1,2)
> m1=mean(x) #样本均值
> m2=var(x) #样本方差
>model=function(x,m1,m2){
+   c(f1=x[1]+x[2]-2*m1,
+     f2=(x[2]-x[1])^2/12-m2)
+ }
>library(rootSolve)
>multiroot(f=model,start=c(0,10),m1=m1,m2=m2)
$root
[1] -0.7523918 10.2523918
$f.root
      f1      f2
-5.153211e-12 1.121688e-09
```

```
$iter
[1] 4
$estim.precis
[1] 5.634204e-10
```

第一行给出的结果即均匀分布的两个参数值[-0.75, 10.25]。

`rootSolve` 在解一些较复杂的方程组时十分方便，本例中的均匀分布比较简单，解方程组是可以得到解析解的，所以其可帮助我们验证上述代码计算的结果。

$$\hat{\theta}_1 = \bar{x} - \sqrt{3}S$$

$$\hat{\theta}_2 = \bar{x} + \sqrt{3}S$$

```
> m1-sqrt(3*m2);m1+sqrt(3*m2)
[1] -0.7523918
[1] 10.25239
```

6.1.2 极大似然估计

极大似然估计法于 1912 年由英国统计学家 R.A.Fisher 提出，其建立在德国数学家 C.F.Gauss 的理论基础上，具有很多优良的性质。极大似然估计利用样本分布密度构造似然函数来求出参数的估计值，相比矩估计更加充分利用了样本信息，因此是目前应用非常广泛的参数估计方法。

为了形象地说明极大似然原理，首先举一个例子。一个常年打猎的猎人和一个“菜鸟”一起外出打猎，发现一只猎物从前方窜过，只听一声枪响，猎物应声倒下，如果我们要推测命中的子弹是谁打的？最直观的思路就是，只需一枪便打中猎物，由于猎人命中的概率肯定大于菜鸟，那么这一枪多半是猎人射中的。

类似的道理，在概率论中我们常常要处理“小球”问题：设甲箱中有 99 个白球，1 个黑球；乙箱中有 1 个白球，99 个黑球。现随机选取一箱，再从中随机抽取一个小球，结果是黑球，由于乙箱中 99%都是黑球，所以我们自然更多地相信这个黑球是取自乙箱的。所以说，极大似然估计法就是要选取这样的数值作为参数的估计值，使所选取的样本在被选的总体中出现的可能性为最大。

极大似然估计既适用于离散分布，也适用于连续分布，不同之处仅在于离散型随机变量使用的函数是分布律 $p(x|\theta)$ ，连续分布使用概率密度函数 $f(x|\theta)$ ，其中 θ 代表未知参数（1 个或多个）。设 X_1, X_2, \dots, X_n 为来自总体 X 的样本，样本观测值为 x_1, x_2, \dots, x_n ，其联合概率函数是 θ 的函数，并用 $L(\theta)$ 表示，即

$$L(\theta) = L(x_1, x_2, \dots, x_n; \theta) = \prod_{i=1}^n p(x_i | \theta) \text{ 或 } \prod_{i=1}^n f(x_i | \theta)$$

称 $L(\theta)$ 为似然函数，对样本的任何观测值 x_1, x_2, \dots, x_n ，若

$$L(\hat{\theta}) = L(x_1, x_2, \dots, x_n; \hat{\theta}) = \sup_{\theta \in \Theta} L(x_1, x_2, \dots, x_n; \theta)$$

则称使 $L(\theta)$ 达到最大的参数值 $\hat{\theta}$ 即为 θ 的极大似然估计值。若 $p(x|\theta)$ 或 $f(x|\theta)$ 关于 θ 可微, 则极大似然估计 $\hat{\theta}$ 可以通过方程 $\frac{\partial L(\theta)}{\partial \theta} = 0$ 得到。但这个方程解起来十分复杂, 尤其似然函数大多是乘积的形式; 而 $\ln(x)$ 是 x 的单调函数, 对数的形式可以将乘积化为加法, 因此参数 θ 的最大似然估计 $\hat{\theta}$ 也可以通过方程 $\frac{\partial \ln[L(\theta)]}{\partial \theta} = 0$ 得到, 后者的求解会简便许多。

(1) 使用 R 中的极值函数计算

同样, R 中计算极值的函数也有多个, 首先在表 6.3 中列出这些函数, 之后将逐一介绍它们的使用方法。

表 6.3 R 中计算极值的函数

函 数	功 能	程序包
optimize()	计算单参数分布的极大似然估计值	stats
optim()	计算多个参数分布的极大似然估计值	stats
nlm()	计算非线性函数的最小值点	stats
nlminb()	非线性最小化函数	stats

1. 函数 optimize()

当分布只包含一个参数时, 我们可以使用 R 中计算极值的函数 optimize() 求极大似然估计值。由于函数来自统计分析程序包 stats, 因此在打开 R 时会自动加载, 其调用格式如下:

```
optimize(f = , interval = , ..., lower = min(interval), upper = max(interval), maximum
= FALSE,
tol = .Machine$double.eps^0.25)
```

其中 f 是似然函数; $interval$ 指定参数的取值范围; $lower/upper$ 分别是参数的下界和上界; $maximum$ 默认为 FALSE, 表示求似然函数的极小值, 若为 TRUE 则求极大值; tol 表示计算的精度。

2. 函数 optim() 和 nlm()

当分布包含多个参数时, 用函数 optim() 或 nlm() 计算似然函数的极大值点。调用格式分别为:

```
optim(par, fn, gr=NULL, ...,
      method=c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN", "Brent"),
      lower = -Inf, upper = Inf, control = list(), hessian = FALSE)
```

par 设置参数的初始值; fn 为似然函数; $method$ 提供了 5 种计算极值的方法。

```
nlm(f, p, ..., hessian = FALSE, typesize = rep(1, length(p)), fscale = 1, print.level = 0,
ndigit = 12, gradtol = 1e-6, stepmax = max(1000 * sqrt(sum((p/typesize)^2)), 1000),
```

```
steptol = 1e-6, iterlim = 100, check.analyticals = TRUE)
```

`nlm` 是非线性最小化函数，仅使用牛顿-拉夫逊算法，通过迭代计算函数的最小值点。一般只需要对前两个参数进行设置：`f` 是需要最小化的函数；`p` 设置参数初始值。

3. 函数 `nlminb()`

在实际应用中，上面这三个基本函数在遇到数据量较大或分布较复杂的计算时，往往很难得到估计结果。因此这里着重介绍另一个优化函数 `nlminb()`。

```
nlminb(start, objective, gradient = NULL, hessian = NULL, ...,
       scale = 1, control = list(), lower = -Inf, upper = Inf)
```

参数 `start` 是数值向量，用于设置参数的初始值；`objective` 指定要优化的函数；`gradient` 和 `hess` 用于设置对数似然的梯度，通常采用默认状态；`control` 是一个控制参数的列表；`lower` 和 `upper` 设置参数的下限和上限，如果未指定，则假设所有参数都不受约束。

`nlminb()` 也是非线性最小化函数，但它使用端口程序（PORT routines）进行约束或无约束的优化，方法上的不同使其计算能力要比前面几个函数强大很多。用 R 进行分布拟合时，最困难的要属“双峰”形状的混合分布，因为涉及参数很多，一般的优化函数都无法得出计算结果，这时 `nlminb()` 的强大就体现出来了。我们使用程序包 MASS 中的数据 `geyser` 来证明这一点，该数据集是一个地质学家记录的美国黄石公园内一个名为 Old Faithful 的喷泉在一年内的喷发数据，其中包括两个变量，分别是泉水持续时间（`eruptions`）和喷发相隔时间（`waiting`）。先简单看一下数据的样子：

```
>library(MASS)
>head(geyser,5)
waiting duration
1      80 4.016667
2      71 2.150000
3      57 4.000000
4      80 4.000000
5      75 4.000000
```

我们要对变量 `waiting` 进行分布拟合，拟合前要通过直方图（如图 6.2 所示）了解数据分布的形态。

```
>attach(geyser)
>hist(waiting,freq=FALSE)
```

从图 6.2 中可以看出，数据有两个峰，像是两个分布叠加在一起的，我们可以猜测分布是两个正态分布的混合，用如下函数来描述

$$f(x) = p \cdot N(x; \mu_1, \sigma_1) + (1 - p) \cdot N(x; \mu_2, \sigma_2)$$

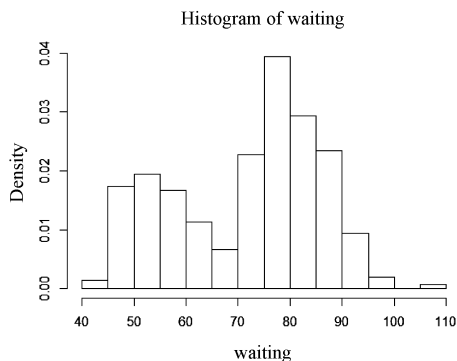


图 6.2 “双峰”数据的直方图

为了正确地描述分布状态，需要估计出函数中的 5 个参数： p 、 μ_1 、 σ_1 、 μ_2 和 σ_2 ，首先写出分布函数的对数似然函数。

$$\begin{aligned}
 l &= \sum_{i=1}^n \log \{pN(x; \mu_1, \sigma_1) + (1-p)N(x; \mu_2, \sigma_2)\} \\
 &= \sum_{i=1}^n \log(f)
 \end{aligned}$$

在 R 中编写对数似然函数时，5 个参数都存放在向量 `para` 中，由于 `nlminb()` 是计算极小值的，因此函数 `function` 中最后返回的是对数似然函数的相反数。

```
>ll=function(para)
+ {
+   f1=dnorm(waiting,para[2],para[3])
+   f2=dnorm(waiting,para[4],para[5])
+   f=para[1]*f1+(1-para[1])*f2
+   ll=sum(log(f))
+   return(-ll)
+ }
```

接下来做参数估计，使用 `nlminb()` 之前最大的要点是确定初始值，初始值越接近真实值，计算的结果才能越精确。我们猜想数据的分布是两个正态的混合，概率 p 直接用 0.5 做初值即可，通过直方图中两个峰对应的 x 轴数值（大概为 50 和 80），就可以将初值设定为 μ_1 和 μ_2 。而概率 p 处于 (0,1) 区间内，参数 σ_1, σ_2 是正态分布的标准差，必须大于 0，所以通过 `lower` 和 `upper` 两个参数进行一定的约束。

```
>geyser.est=nlminb(c(0.5,50,10,80,10),ll,lower=c(0.0001,-Inf,0.0001,-Inf,0.0001),
upper=c(0.9999,Inf,Inf,Inf,Inf))
>options(digits=3)
>geyser.est$par #查看拟合的参数结果
```

```
[1] 0.308 54.203 4.952 80.360 7.508
> p=geyser.est$par[1]
> mu1=geyser.est$par[2];sigma1=geyser.est$par[3]
> mu2=geyser.est$par[4];sigma2=geyser.est$par[5]
> x=seq(40,120)
> #将估计的参数函数代入原密度函数
> f=p*dnorm(x,mu1,sigma1)+(1-p)*dnorm(x,mu2,sigma2)
> hist(waiting,freq=F)
> lines(x,f) #画出拟合曲线
```

绘制结果如图 6.3 所示。

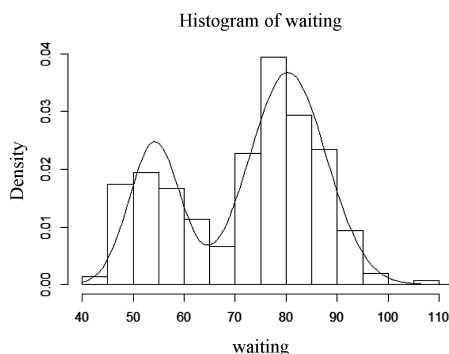


图 6.3 混合分布的拟合效果

从图形效果来看, `nlminb()` 可以很好地对混合分布作极大似然估计, 即使参数有 5 个之多, 也能准确地给出估计结果。

(2) 使用极大似然估计函数 `maxLik()` 计算

除了以上几个基本函数外, R 中有专门的程序包用于作极大似然估计, 笔者最常用的是 `maxLik`, 它不仅使用简单, 而且即使在初始值偏离较大的情况下, 计算得到的估计值也非常接近真值, 计算准确度较高。而上面介绍的两个函数, 若参数初始值设定得不准确, 很有可能会无法计算出结果。程序包 `maxLik` 中同名的函数 `maxLik()` 可以直接计算极大似然估计值, 调用格式如下:

```
maxLik(logLik, grad = NULL, hess = NULL, start, method, constraints=NULL, ...)
```

其中的参数意义与 `nlminb()` 中的比较类似。 `logLik` 是对数似然函数, 这里要特别注意, `logLik` 引用的函数必须是经过对数调整后的; `grad` 和 `hess` 用于设置对数似然的梯度, 通常不需要进行设置, 采用默认值 `NULL` 即可; `start` 是一个数值向量, 设置参数的初始值; `method` 选择求解最大化的方法, 包括“牛顿-拉夫逊”、“BFGS”、“BFGSR”、“BHHH”、“SANN”和“Nelder-Mead”, 如果不设置, 将自动选择一个合适的方法; `constraints` 指定对似然估计的约束。

极大似然估计不仅适用于连续分布, 也适用于离散分布。所以下面我们使用第一节中给出的损失次数的数据 `num`, 采用两参数的负二项分布做极大似然估计, 具体说明离散分布的拟合。负

二项分布的参数为 (r, β) ，概率函数为：

$$p_k = \frac{\Gamma(k+r)}{\Gamma(r)\Gamma(k+1)} p^r (1-p)^k, \quad k=0,1,2,\dots$$

$$p = \frac{1}{1+\beta}$$

编写 R 程序时首先要写出对数似然函数 `loglik`，用到 R 中的负二项函数 `dnbinom()`，它的参数是 `r`、`p`。如果要估计 β 的值，应当转换一下形式。使用 `maxLik` 函数后将得到一个列表，包含很多信息，其中的 `estimate` 为两个参数的估计值。

```
> num=c(rep(0:5,c(1532,581,179,41,10,4)))
> install.packages("maxLik")
> loglik=function(para){
+   f=dnbinom(num,para[1],1/(1+para[2]))#注意第二个参数不是 beta，是 prob
+   ll=sum(log(f))
+   return(ll)
+ }
> library(maxLik)
> para=maxLik(loglik,start=c(0.5,0.4))$estimate #极大似然估计
> r=para[1];beta=para[2]
```

参数的检验将在下文介绍，这里我们仍通过图形来观察估计的效果，比较损失次数的样本值和估计值。

```
> l=length(num)
> nbinomnum=dnbinom(0:5,r,1/(1+beta))*l;nbinomnum
[1] 1530.116467 588.080543 170.655271 44.166955 10.737733 2.509432
> plot(0:5,nbinomnum,ylim=c(0,1600)) #画图比较
> points(0:5,nbinomnum,type="p",col=2)
> legend(3,1000,legend=c("num","poisson"),col=1:2,lty=1)
```

绘制结果如图 6.4 所示。

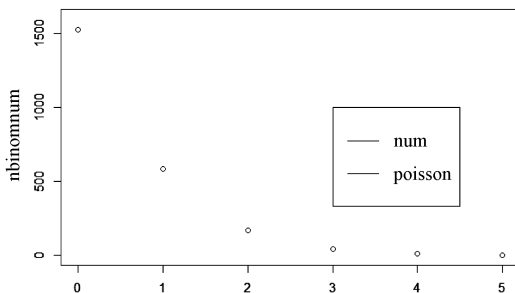


图 6.4 损失次数的极大似然估计

由图 6.4 可以看出，负二项分布的极大似然估计效果非常好，估计值与样本值几乎完全重合，因此我们可以得出结论，损失次数服从负二项分布。

6.2 单正态总体的区间估计

6.1 节介绍的点估计值只是估计量的一个近似值，但是我们不禁会考虑：这种点估计的精确性如何？有多大的置信度？点估计本身是无法解决这些问题的，因为它没有反映出估计的误差范围有多大，这些正是区间估计要讨论的问题。区间估计是重要的统计推断形式，其引入了置信度的概念，按照一定的正确度和精确度要求，利用抽取的样本构造出适当的区间，作为总体分布的未知参数真值的估计范围。例如人们常说有百分之多少的把握保证某值在某个范围内，即是区间估计的最简单的应用。若满足式子

$$P\{\hat{\theta}_1(X_1, X_2, \dots, X_n) < \theta < \hat{\theta}_2(X_1, X_2, \dots, X_n)\} = 1 - \alpha$$

则称区间 $(\hat{\theta}_1, \hat{\theta}_2)$ 是参数 θ 的置信度为 $1 - \alpha$ 的置信区间，也就是说此区间包含 θ 的可能性（置信度）是 $1 - \alpha$ 。

首先我们来讨论单个正态总体的情况， $X \sim N(\mu, \sigma^2)$ ， X_1, X_2, \dots, X_n 是来自正态总体的一个样本， \bar{X} 是样本均值， S^2 是样本方差。

6.2.1 均值 μ 的区间估计

单个正态总体均值的估计，分别从总体方差 σ^2 已知和未知两种情形入手。

(1) σ^2 已知

由中心极限定理可知 $\bar{X} \sim N(\mu, \frac{\sigma^2}{n})$ ，因此有 $Z = \frac{\bar{X} - \mu}{\sigma / \sqrt{n}} \sim N(0, 1)$ 。

再由概率 $P(-z_{\alpha/2} < Z < z_{\alpha/2}) = 1 - \alpha$ 可得

$$p\left(\bar{X} - \frac{\sigma}{\sqrt{n}} z_{\alpha/2} < \mu < \bar{X} + \frac{\sigma}{\sqrt{n}} z_{\alpha/2}\right) = 1 - \alpha$$

其中， $z_{\alpha/2}$ 是标准正态分布 $N(0, 1)$ 上的 $\alpha/2$ 分位点，由此我们可以得到关于均值 μ 、置信水平为 $1 - \alpha$ 的双侧置信区间为

$$\left[\bar{X} - \frac{\sigma}{\sqrt{n}} z_{\alpha/2}, \bar{X} + \frac{\sigma}{\sqrt{n}} z_{\alpha/2} \right]$$

在 R 中实现单正态总体均值的区间估计（方差已知）可以有三种实现方式。首先，我们知道

R 中没有计算方差已知时均值置信区间的内置函数，需要自己编写，代码也比较简单：

```
conf.int=function(x, sigma, alpha){
  mean=mean(x)
  n=length(x)
  z=qnorm(1-alpha/2, mean=0, sd=1, lower.tail=TRUE)
  c(mean-sigma*z/sqrt(n), mean+sigma*z/sqrt(n))
}
```

其中 x 为数据样本； σ 是已知总体的标准差； α 表示显著性水平。通常我们作区间估计时，都会估计出双侧的置信区间，因为它为待估参数提供了上下限两个参考值。但如果要估计单侧的置信区间，理论上与双侧相同，只需要使用标准正态分布的 α 分位点即可，编写函数时也可做同样变动即可。

事实上，随着更多统计程序包的开发，现在基本统计和数据分析程序包 BSDA (Basic Statistics and Data Analysis) 中已经提供了函数 `z.test()`，它可以对基于正态分布的单样本和双样本进行假设检验、区间估计，其使用方法如下：

```
z.test(x, y = NULL, alternative = "two.sided", mu = 0, sigma.x = NULL,
sigma.y = NULL, conf.level = 0.95)
```

其中， x 和 y 为数值向量，默认 $y=NULL$ ，即进行单样本的假设检验；`alternative` 用于指定所求置信区间的类型，默认为 `two.sided`，表示求双尾的置信区间，若为 `less` 则求置信上限，为 `greater` 求置信下限； μ 表示均值，它仅在假设检验中起作用，默认为 0； $\sigma.x$ 和 $\sigma.y$ 分别指定两个样本总体的标准差；`conf.level` 指定区间估计时的置信水平。

第三个函数是程序包 `UsingR` 中的函数 `simple.z.test()`，它专门用于对方差已知的样本均值进行区间估计，与 `z.test()` 的不同点在于它只能进行置信区间估计，而不能实现 Z 检验。`simple.z.test()` 的使用方法如下：

```
simple.z.test(x, sigma, conf.level=0.95)
```

其中， x 是数据向量； σ 是已知的总体标准差；`conf.level` 指定区间估计的置信度，默认为 95%。

举例说明，首先从均值为 10、标准差为 2 的总体中抽取 20 个样本，因此这是一个方差已知的正态分布样本。计算置信水平为 95% 时 x 的置信区间，首先调用自行编写的函数 `conf.int()`，输入显著性水平参数 $\alpha = 0.05$ 。

```
> set.seed(111) #设定随机种子
> x=rnorm(20,10,2)
> conf.int(x,2,0.05)
[1] 8.416051 10.169096
```

用函数 `z.test()` 也可以直接得到这一结果，函数 `z.test()` 计算后将输出很多信息，如果只要区间

估计的结果, 则用符号 “\$” 选取 `conf.int` 的内容:

```
> library("BSDA")
> z.test(x, sigma.x=2)$conf.int
[1] 8.416051 10.169096
attr(,"conf.level")
[1] 0.95
```

第三种实现方法是函数 `simple.z.test()`, 它可以直接得到区间估计结果。

```
> library("UsingR", lib.loc="D:/R-3.0.2/R-3.0.2/library")
Loading required package: MASS
> simple.z.test(x,2)
[1] 8.416051 10.169096
```

三种方法的结果均显示, 该样本的 95% 置信区间为 [8.42, 10.17]。

对于非正态总体, 当样本量很大时也可以使用上面的函数做均值的区间估计。根据中心极限定理, 对于足够大的 n , 近似成立

$$\frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \sim N(0,1)$$

如果方差 σ^2 未知, 则可以使用样本方差 S^2 代替。在置信水平 $1-\alpha$ 下导出均值的置信区间为

$$\left[\bar{X} - \frac{S}{\sqrt{n}} z_{\alpha/2}, \bar{X} + \frac{S}{\sqrt{n}} z_{\alpha/2} \right]$$

这与正态总体的区间估计形式一致, 因此也可以使用函数 `conf.int()`。

(2) σ^2 未知

总体方差未知时, 用 t 分布的统计量来替代 z , 方差也要由样本方差 s^2 代替, 所以此时的置信区间为

$$\left[\bar{X} - \frac{s}{\sqrt{n}} t_{\alpha/2}(n-1), \bar{X} + \frac{s}{\sqrt{n}} t_{\alpha/2}(n-1) \right]$$

在 R 中实现方差未知的区间估计非常容易。我们知道, 参数估计与假设检验有着密不可分的联系, 所以调用 t 检验的函数就可以直接求出置信区间了, `t.test()` 的调用格式如下。这里先介绍用它计算区间估计的方法, 假设检验相关的部分将在下文中详细介绍。

```
t.test(x, y = NULL, alternative = c("two.sided", "less", "greater"),
mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95, ...)
```

其中, x 为样本数据; 若 x 和 y 同时输入, 则做双样本 t 检验; `alternative` 用于指定所求置信区间的类型, 默认为 `two.sided`, 表示求双尾的置信区间, 若为 `less` 则求置信上限, 为 `greater` 求置信下限; `mu` 表示均值, 其仅在假设检验中起作用, 默认为 0。

仍使用上例中的向量 x , 假设总体方差未知时, 用函数 `t.test()` 计算置信区间后将输出很多信息, 如果只要区间估计的结果, 则用符号 “\$” 选取 `conf.int` 的内容:

```
>t.test(x)$conf.int
[1] 8.19 10.47
attr(,"conf.level")
[1] 0.95
```

总体方差未知时得到 x 的置信区间为 [8.19, 10.47], 比方差已知的区间要更宽, 这个结果十分合理, 因为方差未知时我们可用的信息更少, 估计结果相对不那么精确。

6.2.2 方差 σ^2 的区间估计

估计均值时, 我们分方差已知和方差未知两种情况分别进行了讨论。类似地, 方差的区间估计也根据均值已知和未知两种情形讨论。

(1) μ 已知

当 μ 已知, 根据 σ^2 的极大似然估计 $\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (X_i - \mu)^2$ 来推导置信区间。首先, 根据卡方分布的定义

$$\frac{n\hat{\sigma}^2}{\sigma^2} = \sum_{i=1}^n (X_i - \mu)^2 / \sigma^2 \sim \chi^2(n)$$

所以由 $P\left\{\chi_{1-\alpha/2}^2(n) \leq \frac{n\hat{\sigma}^2}{\sigma^2} \leq \chi_{\alpha/2}^2(n)\right\} = 1 - \alpha$ 就可以得到 σ^2 的置信水平为 $1 - \alpha$ 的双侧置信区间

$$\left[\frac{n\hat{\sigma}^2}{\chi_{\alpha/2}^2(n)}, \frac{n\hat{\sigma}^2}{\chi_{1-\alpha/2}^2(n)} \right]$$

其中, $\chi_{1-\alpha/2}^2(n)$ 和 $\chi_{\alpha/2}^2(n)$ 分别为自由度为 n 的卡方分布上 $1 - \alpha/2$ 和 $\alpha/2$ 分位点。

(2) μ 未知

当 μ 未知时, σ^2 的极大似然估计为样本方差, 即 $S^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2$ 。由于

$$\chi^2 = \frac{(n-1)S^2}{\sigma^2} \sim \chi^2(n-1)$$

因此，有 $P\left\{\chi_{1-\alpha/2}^2(n-1) \leq \frac{(n-1)S^2}{\sigma^2} \leq \chi_{\alpha/2}^2(n-1)\right\} = 1-\alpha$ ，从而得到均值未知时方差的

置信区间（置信水平为 $1-\alpha$ ）为

$$\left[\frac{(n-1)S^2}{\chi_{\alpha/2}^2(n-1)}, \frac{(n-1)S^2}{\chi_{1-\alpha/2}^2(n-1)} \right]$$

在 R 中没有直接计算方差的置信区间的函数，我们可以把上面两种情况写在一个函数里，通过一个 if 语句进行判断，只要是方差的区间估计，都调用这个函数即可。在 R 中写函数时，参数可以事先设定一个初值，例如设 `mu=Inf`，代表均值未知的情况，调用函数时如果没有特殊说明 `mu` 的值，将按照均值未知的方法计算；如果均值已知，在调用函数时应该对 `mu` 重新赋值。

```
var.conf.int=function(x,mu=Inf,alpha){
  n=length(x)
  if(mu<Inf){
    s2=sum((x-mu)^2)/n
    df=n
  }
  else{
    s2=var(x)
    df=n-1
  }
  c(df*s2/qchisq(1-alpha/2,df),df*s2/qchisq(alpha/2,df))
}
```

在实际分析中，均值 μ 基本都是未知的情形，例如计算样本 X 的方差的置信区间时，若均值未知，调用函数时就不需要指定 `mu` 的值了。

```
> var.conf.int(x,alpha=0.05)
[1] 3.4112.58
```

计算得到总体方差的置信区间为[3.41, 12.58]，置信水平是 95%。

6.3 两正态总体的区间估计

单个总体的区间估计针对总体的均值和方差，当有两个总体的时候，就要对它们的参数进行对比了，这在实际应用中很常见。比如，我们要考察一项新技术对提高产品质量是否有成效，则把新技术实施前后的产品质量指标看成两个正态总体，这时我们所考察的问题，就转化为检验这两个正态总体的均值比较的问题。

设 X 与 Y 是两个独立的总体, $X \sim N(\mu_1, \sigma_1^2), Y \sim N(\mu_2, \sigma_2^2)$, X_1, X_2, \dots, X_{n_1} 是来自总体 X 的样本, Y_1, Y_2, \dots, Y_{n_2} 是来自总体 Y 的样本, 它们的均值分别为 \bar{X} 和 \bar{Y} , 方差分别为 S_1^2 和 S_2^2 。

6.3.1 均值差 $\mu_1 - \mu_2$ 的区间估计

根据两样本总体方差是否已知、是否相等, 这里要分三种情况讨论。

(1) 两个总体的方差 σ_1^2 、 σ_2^2 已知

对于两个正态总体, 当它们的方差都已知时, 根据正态分布的性质有

$$\bar{X} - \bar{Y} \sim N\left(\mu_1 - \mu_2, \frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}\right)$$

类似于单正态总体区间估计的推导, 可以得到 $\mu_1 - \mu_2$ 的置信水平为 $1 - \alpha$ 的双侧置信区间, 这里省略推导过程, 直接给出结果:

$$\left[\bar{X} - \bar{Y} - z_{\alpha/2} \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}, \bar{X} - \bar{Y} + z_{\alpha/2} \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}} \right]$$

在 R 中编写计算置信区间的函数 `twosample.ci()` 如下, 输入参数为样本 x 、 y 、置信度 α 和两个样本的标准差。

```
twosample.ci=function(x,y,alpha,sigma1,sigma2){
  n1=length(x);n2=length(y)
  xbar=mean(x)-mean(y)
  z=qnorm(1-alpha/2)*sqrt(sigma1^2/n1+sigma2^2/n2)
  c(xbar-z,xbar+z)
}
```

前面介绍的 Z 检验函数 `z.test()` 可以在两总体方差已知的情况下, 计算两总体均值差的置信区间, 分别用参数 `sigma.x` 和 `sigma.y` 来说明已知的标准差数值即可。

介绍过理论后, 我们以 Bamberger's 百货公司为案例做一个实证分析。Bamberger's 是一家为社区提供大众性商品的零售商店, 你几乎可以从这里买到诸如香水到链锯之类的任何东西, 长期以来, Bamberger's 的顾客服务有着极好的口碑。为了努力维持商店的良好声誉, 公司实施了将营业时间延长至夜间的计划。

以 Bamberger's 延长营业时间前后 27 个典型周的销售数据为例 (以万元为单位), 计算这两个样本均值差的区间估计, 从而可以看出计划实施后的效果。首先查看数据的基本类型, 并绘制直方图对比。

```

>sales=read.table("d:/data/sales.txt",header=T)
>head(sales)
prior  post
1  67.90  86.10
2  76.12  71.13
3  68.64 116.25
4  74.94 102.60
5  63.32  97.51
6  50.43  65.39
>attach(sales)
>par(mfrow=c(1,2))
>hist(prior) #分别绘制计划前后销售额的直方图
>hist(post)

```

绘制结果如图 6.5 所示。

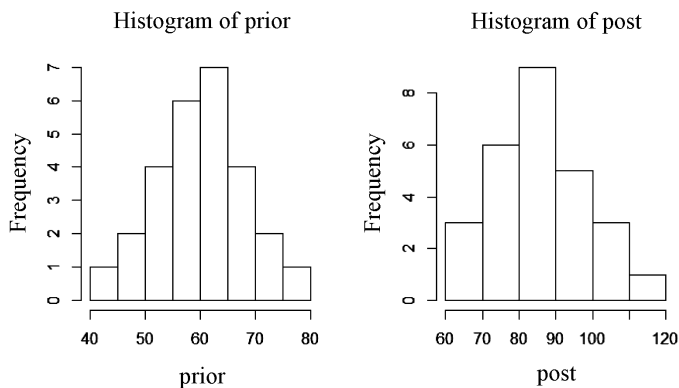


图 6.5 Bamberger's 营业额的直方图

从直方图可以看出，销售额样本大致呈正态分布，假设已知计划实施前后的总体标准差分别为 8 和 12，调用上面写好的函数，计算样本均值差 $\mu_1 - \mu_2$ 在置信水平为 $1 - \alpha$ 下的置信区间。

```

>twosample.ci(post,prior,alpha=0.05,8,12)
[1] 19.10298 29.98295

```

直接使用程序包 DBSA 中的函数 `z.test()` 也可以得到相同的结果：

```

> z.test(post,prior,sigma.x=8,sigma.y=12)$conf.int
[1] 19.10298 29.98295
attr(,"conf.level")
[1] 0.95

```

区间估计的结果是，Bamberger's 公司延长营业时间后周营业额明显增加，增加额的范围是 [19.10, 29.98]。

(2) 两个总体的方差 σ_1^2 、 σ_2^2 未知但相等

当条件是方差 σ_1^2 、 σ_2^2 未知但相等时，可以得到服从 t 分布的统计量

$$T = \frac{(\bar{X} - \bar{Y}) - (\mu_1 - \mu_2)}{\sqrt{\left(\frac{1}{n_1} + \frac{1}{n_2}\right) S^2}} \sim t(n_1 + n_2 - 2)$$

其中

$$S^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}$$

由于 $P(-t_{\alpha/2}(n_1 + n_2 - 2) < T < t_{\alpha/2}(n_1 + n_2 - 2)) = 1 - \alpha$ ，解不等式可以得到 $\mu_1 - \mu_2$ 的置信水平为 $1 - \alpha$ 的双侧置信区间为

$$\bar{X} - \bar{Y} \pm t_{\alpha/2}(n_1 + n_2 - 2) \sqrt{\left(\frac{1}{n_1} + \frac{1}{n_2}\right) S^2}$$

正如计算单正态总体均值的置信区间，R 中的函数 `t.test()` 还可以用来求两总体均值差的置信区间，由于总体方差相等，需要将其中的参数 `var.equal` 设为 `TRUE`。在 Bamberger's 公司的案例中，如果延长营业时间前后总体的方差未知但相等，我们就可以通过 `t.test()` 直接计算置信区间了，注意用美元符号 “\$” 选取检验结果中的项目 `conf.int`。

```
>t.test(post,prior,var.equal=TRUE)$conf.int
[1] 18.66541 30.42051
attr(,"conf.level")
[1] 0.95
```

由计算结果同样可以得到结论：Bamberger's 公司延长营业时间后周营业额明显增加，在 0.95 的置信水平下，营业额增加的置信区间是 [18.67, 30.42]。

(3) 两个总体的方差 σ_1^2 、 σ_2^2 未知且不等

当两总体的方差未知并且 $\sigma_1^2 \neq \sigma_2^2$ 时会近似得到

$$T = \frac{(\bar{X} - \bar{Y}) - (\mu_1 - \mu_2)}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}} \sim t(\nu)$$

用样本方差代替总体方差，得到 ν 的估计值

$$\hat{\nu} = \left(\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2} \right)^2 \bigg/ \left(\frac{(S_1^2/n_1)^2}{n_1 - 1} + \frac{(S_2^2/n_2)^2}{n_2 - 1} \right)$$

所以我们可以近似地认为 $T \sim t(\hat{\nu})$ ，由此得到 $\mu_1 - \mu_2$ 的置信水平为 $1 - \alpha$ 的双侧置信区间为

$$(\bar{X} - \bar{Y}) \pm t_{\alpha/2}(\hat{\nu}) \sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}$$

显然，第三种情况比较复杂，R 中也没有直接的函数可用，仍需要手动写出一个函数 `twosample.ci2()`。

```
twosample.ci2=function(x,y,alpha){
  n1=length(x);n2=length(y)
  xbar=mean(x)-mean(y)
  S1=var(x);S2=var(y)
  nu=(S1/n1+S2/n2)^2/(S1^2/n1^2/(n1-1)+S2^2/n2^2/(n2-1))
  z=qt(1-alpha/2,nu)*sqrt(S1/n1+S2/n2)
  c(xbar-z,xbar+z)
}
```

在实际分析中，两总体的方差未知且不等是最常见的情况，在 Bamberger's 公司的案例中，如果延长营业时间前后的方差未知并且不相等，就要通过上面编写的函数计算样本均值差的置信区间。

```
>twosample.ci2(post,prior,0.05)
[1] 18.63821 30.44771
```

相比之前，营业时间延长后的样本均值明显增加，两样本均值差的置信区间为[18.64, 30.45]。由于方差未知，在做区间估计时可利用的信息更少，因此在相同置信水平下，这时估计得到的置信区间相对更宽一些。

6.3.2 两方差比 σ_1^2 / σ_2^2 的区间估计

在实际分析中，总体均值通常是未知的，尤其涉及到两个总体时。比较两个总体的方差，采用的是“比”的形式，而不是差值，这与统计量的构造有关。由于

$$\frac{(n_1 - 1)S_1^2}{\sigma_1^2} \sim \chi^2(n_1 - 1), \quad \frac{(n_2 - 1)S_2^2}{\sigma_2^2} \sim \chi^2(n_2 - 1)$$

总体 X 和 Y 独立，所以 S_1^2 和 S_2^2 相互独立。两个卡方分布的随机变量的比值服从 F 分布，故

$$F = \frac{S_1^2/\sigma_1^2}{S_2^2/\sigma_2^2} \sim F(n_1 - 1, n_2 - 1)$$

所以对于给定的置信水平 $1-\alpha$ ，有不等式

$$P\left\{F_{1-\alpha/2}(n_1-1, n_2-1) \leq \frac{S_1^2/\sigma_1^2}{S_2^2/\sigma_2^2} \leq F_{\alpha/2}(n_1-1, n_2-1)\right\} = 1-\alpha$$

其中， $F_{\alpha/2}(n_1-1, n_2-1)$ 表示自由度为 (n_1, n_2) 的 F 分布上 $\alpha/2$ 的分位点。由不等式变形得到 σ_1^2/σ_2^2 的置信区间为

$$\left[\frac{S_1^2/S_2^2}{F_{\alpha/2}(n_1-1, n_2-1)}, \frac{S_1^2/S_2^2}{F_{1-\alpha/2}(n_1-1, n_2-1)} \right]$$

方差比的区间估计与方差的假设检验密不可分，所以 R 中的函数 `var.test()` 可以用来直接计算两正态总体方差比的置信区间，调用格式如下：

```
var.test(x, y, ratio = 1, alternative = c("two.sided", "less", "greater"), conf.level = 0.95, ...)
```

其中的参数设置与上文中的 t 检验函数一致，`ratio` 指定原假设中方差比的值，通过 `alternative` 设置单侧区间或双侧。

仍以 Bamberger's 公司的数据为例，如果想要考察延长营业时间前后，周营业额的波动情况是否一致，这就转化为两方差比估计的问题了。

```
>var.test(prior, post)$conf.int
[1] 0.1772458 0.8534348
attr(,"conf.level")
[1] 0.95
```

通过函数 `var.test()` 计算后的结果可以看出，两样本方差比在 95% 置信水平下的区间估计为 $[0.1772, 0.8534]$ ，这说明延长营业额后，周营业额的波动性变大。

6.4 关于比率的区间估计

从小学做数学题开始，我们就经常见到关于比例或比率的问题，例如某个学校的男女比例，某次考试的及格率、优秀率，某种产品生产后的合格率、废品率，甚至某个电视节目的收视率，等等。这说明在实际分析中，我们常常要估计在总体中具有某种特性的个体占总体的比例。比率的点估计使用第一节介绍的方法即可，这里介绍计算比率的近似区间估计方法。

在一个容量为 n 的样本中，设 x 为具有某种特征的样本个数，那么样本比例为 x/n 。比率的估计要建立在大量样本量的基础上，这是因为当 n 足够大时，根据中心极限定理可知样本比率的抽样分布近似于正态分布，从而可以使用正态分布的规律进行有关总体比率的估计。对于总体比率估计量 $\hat{p} = x/n$ 而言，有

$$Z = \frac{\hat{p} - p}{\sqrt{\hat{p}(1 - \hat{p})/n}} \sim N(0,1)$$

根据 $P(-Z_{\alpha/2} < Z < Z_{\alpha/2}) = 1 - \alpha$ ，解不等式可以得到总体比率 p 在置信水平 $1 - \alpha$ 下的置信区间

$$\left(\hat{p} - z_{\alpha/2} \sqrt{\hat{p}(1 - \hat{p})/n}, \hat{p} + z_{\alpha/2} \sqrt{\hat{p}(1 - \hat{p})/n} \right)$$

比率的估计在 R 中实现起来也比较简单，函数 `prop.test()` 可以直接完成对 p 的估计和检验，其调用格式为

```
prop.test(x, n, p = NULL, alternative = c("two.sided", "less", "greater"), conf.level = 0.95,
          correct = TRUE)
```

其中，参数 x 为具有某种特征的样本个数； n 为样本量； p 设置假设检验时原假设的比率值；`correct` 是逻辑值，设置是否应用 Yates 连续性修正，默认为 TRUE。

下面举例说明比率的区间估计。某市为了解居民住房情况，抽查了 $n=2000$ 户家庭，其中人均不足 5 平方米的困难户有 $x=214$ 个，通过样本信息计算该市困难户比率 p 的置信区间（置信度为 0.95）。

```
>prop.test(214,2000)

1-sample proportions test with continuity correction

data: 214 out of 2000, null probability 0.5
X-squared = 1234.021, df = 1, p-value < 2.2e-16
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.09396256  0.12157198
sample estimates:
p
0.107
```

比率检验函数的计算结果表明，在 95% 的置信水平下，困难户比率的区间估计为 [0.0940, 0.1216]，而最后一行的 p 值给出的是点估计，该市困难户比率为 0.107。

事实上，当样本数足够多时， x 服从超几何分布，上面我们用的是正态分布近似，但当抽样比很小时还可以用二项分布来近似，这时用到的函数是二项式检验 `binom.test()`，其调用格式如下，内部参数与 `prop.test()` 一致。在上例中，如果该市的总人口数较大，那么抽样比很小，就应当用二项分布近似。

```
binom.test(x, n, p = 0.5, alternative = c("two.sided", "less", "greater"), conf.level = 0.95)
>binom.test(214,2000)
```

```
Exact binomial test

data: 214 and 2000
number of successes = 214, number of trials = 2000, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.09378632  0.12137786
sample estimates:
probability of success
      0.107
```

二项式检验的结果表明，在 95% 的置信水平下，困难户比率的区间估计为 [0.0938, 0.1214]，这与修正正态近似的结果非常接近，点估计值仍为 0.107。

第 7 章

假设检验及 R 实现

参数估计与假设检验是统计推断的两个组成部分，都是利用样本信息对总体进行某种推断，只是角度不同。参数估计仅仅是一种数理统计方法，估计的参数值是对总体参数或分布形式的一种假设，然而它是否准确地描述了总体的分布呢？这时需要假设检验来构造合适的统计量，利用样本信息对所提供的假设进行检验，从而判断是否接受假设。假设检验采用逻辑上的反证法，依据统计上的小概率原理，判断估计数值与总体真实值之间是否存在显著差异。

本章将分别从正态总体的检验以及非参数的检验方面，重点介绍如何运用 R 中的函数完成。

7.1 假设检验概述

对总体参数的具体数值所作的陈述，称为假设；再利用样本信息判断假设是否成立，这整个过程称为假设检验。假设检验可以应用于多种领域，我们首先通过几个例子来说明：

- 判断当前股票价格指数的走势是否正常，我们可以根据过去长期观察得到的平均水平和变异情况作为总体，推断当前股价水平与历史平均水平是否在一定范围内保持一致。
- 某百货公司实施了新的销售计划，管理者十分关心新措施的效果，这就需要用假设检验来判断，经过改革后，公司的营业额是否有显著的提高。
- 在医学统计上，常遇到这样的问题：对于某种疾病，分别采用西药和中西药结合两种疗法，实验中包含两组样本，推断两组疗效有无显著差别。

以上几个例子都是假设检验问题。那么假设检验基于什么原理？应该如何提出假设？怎样构造统计量判断假设是否成立呢？

7.1.1 理论依据

假设检验采用的逻辑推理方法是反证法：为了检验一个假设是否成立，先假定它是成立的，我们称之为“原假设”，然后根据抽样原理和样本信息，观察由这个原假设而导致的结果是否合理。如果不合理，则说明原假设是不正确的，从而得出拒绝原假设的结论；如果结果合理，则不能拒绝原假设。

假设检验之所以可行，其理论背景是小概率理论。小概率事件在一次试验中几乎是不可能发生的，但是它一旦发生，我们就有理由拒绝原假设；反之，小概率事件没有发生，则认为原假设是合理的。这个小概率的标准由研究者事先确定，即以所谓的显著性水平 $\alpha (0 < \alpha < 1)$ 作为小概率的界限， α 的取值与实际问题的性质相关，通常我们取 $\alpha = 0.1$ 、 0.05 或 0.01 。因此，假设检验也称为显著性检验。

需要注意的是，假设检验是带有概率性质的反证法，并非严格的逻辑证明。因此假设检验所基于的样本信息毕竟只是总体的一部分，不能完全代表总体，而是在一定概率（置信度）下来推断总体特征。我们通常把 $1 - \alpha$ 称为置信水平，即对推断结果的把握程度、可靠性。

7.1.2 检验步骤

了解假设检验的基本理论后，我们就可以按照以下几个步骤进行检验了。

(1) 提出假设

首先对检验的未知数 θ 作一个单侧或双侧的假设，假设的选择标准是：通常把研究者要证明的陈述作为备择假设。原假设和备择假设必须是互斥事件，等号必须出现在原假设中。有三种情况：双侧检验、左侧检验和右侧检验。如表 7.1 所示。

表 7.1 原假设与备择假设

假设	双侧检验	单侧检验	
		左侧检验	右侧检验
H_0	$\theta = \theta_0$	$\theta \geq \theta_0$	$\theta \leq \theta_0$
H_1	$\theta \neq \theta_0$	$\theta < \theta_0$	$\theta > \theta_0$

(2) 确定检验统计量，计算统计量的值

假设检验和参数估计在统计量的选择上是一致的。一般对总体的均值、方差和比例进行假设检验，在不同的已知条件下分别采用 Z 统计量、 t 统计量和 χ^2 统计量，例如检验总体均值时可如图 7.1 所示选择。

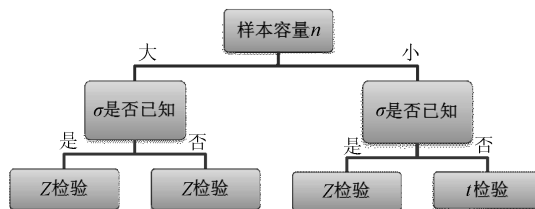


图 7.1 统计量的选择

(3) 规定显著性水平，建立检验规则

显著性水平是原假设为真时拒绝原假设的概率，以 α 表示，常取值为 0.05、0.01 等。在给定的显著性水平和已知的总体分布下，我们可以查表得到相应的临界值，它是划分拒绝域和接受域的数值。拒绝域是检验统计量取值的一个集合，当根据样本计算的统计量属于该集合时，拒绝原假设。

(4) 作出统计决策

最后一步判断是否接受原假设，可以使用两种规则：临界值规则和 P 值规则。

- 临界值规则

双侧检验：| 统计量 | > 临界值时，拒绝 H_0

左侧检验：统计量 ≤ 临界值时，拒绝 H_0

右侧检验：统计量 > 临界值时，拒绝 H_0

- P 值规则

在一个假设检验问题中，拒绝原假设 H_0 的最小显著性水平称为检验的 P 值。 P 值可以告诉我们，如果原假设是正确的话，我们得到目前这个样本统计值的可能性有多大，如果这个可能性很小，就应该拒绝原假设。也就是说， P 值越小，拒绝 H_0 的可能性越大。在显著性水平 α 下， P 值规则为：如果 $P \leq \alpha$ ，则拒绝 H_0 ；如果 $P > \alpha$ ，则不拒绝原假设。

注意：当不能拒绝原假设时，我们不说“接受原假设”，因为没有证据可以说明原假设是真的。

7.1.3 两类错误

显然，我们希望通过假设检验得到正确的判断，即若 H_0 本身是真的，则不拒绝；若 H_0 不正确，则拒绝它。但事实上，假设检验是根据有限的样本信息来推断总体特征的，由于样本的随机性，在推断时不免会犯错误。通常我们所犯的错误有两类：称为第一类错误和第二类错误。第一类错误是否定了真实的原假设，即“弃真”，犯第一类错误的概率就是之前提到的显著性水平 α 。

$$\alpha = P\{\text{拒绝}H_0 | H_0\text{为真}\}$$

通过控制显著性水平 α 可以控制犯第一类错误的概率。而第二类错误是接受了错误的原假设，发生概率用 β 表示，即

$$\beta = P\{\text{不拒绝 } H_0 | H_0 \text{ 为假}\}$$

图 7.2 给出了犯两类错误的概率在分布图上的示例。假设检验中的结论及其后果有以下四种情况，见表 7.2。

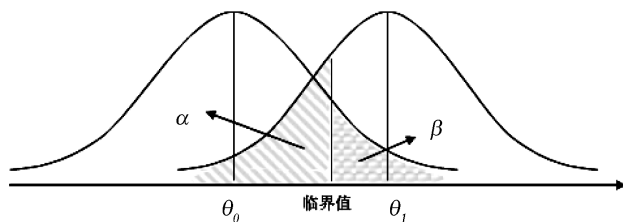


图 7.2 犯两类错误的概率图示

表 7.2 假设检验的四种结果

决策结果	实际结果	
	H_0 为真	H_0 为假
不拒绝 H_0	正确决策	第二类错误 β
拒绝 H_0	第一类错误 α	正确决策

7.2 单正态总体的检验

实际中的很多现象都可以用正态分布去近似地描述，这对应于第 6 章单正态总体的区间估计，我们首先讨论单个正态总体的假设检验问题。假设总体 $X \sim N(\mu, \sigma^2)$ ， X_1, X_2, \dots, X_n 是来自正态总体的一个样本， \bar{X} 是样本均值， S^2 是样本方差。

单正态总体的假设检验组要分情况讨论，不同条件下需要选取不同的检验方法，首先我们将这些方法总结在表 7.3 中，然后再逐一展开介绍。

表 7.3 单正态总体的假设检验方法

检验对象	条件	假设检验方法	函数
均值 μ	σ^2 已知	Z 检验	z.test() (程序包 BSDA)
均值 μ	σ^2 未知	t 检验	t.test() (R 自带函数)
方差 σ^2	μ 已知	卡方检验	chisq.var.test() (自己编写)
方差 σ^2	μ 未知		

7.2.1 均值 μ 的检验

与参数估计一样，单个正态总体均值的假设检验，也分别从总体方差 σ^2 已知和未知两种情形入手。

首先，提出原假设和备择假设

双侧检验： $H_0: \mu = \mu_0$, $H_1: \mu \neq \mu_0$

单侧检验： $H_0: \mu \leq \mu_0$, $H_1: \mu > \mu_0$ 或 $H_0: \mu \geq \mu_0$, $H_1: \mu < \mu_0$

(1) σ^2 已知

当 H_0 为真时，即 $\mu = \mu_0$ ，方差已知设为 $\sigma^2 = \sigma_0^2$ ，检验统计量为服从标准正态分布的 Z 统计量

$$Z = \frac{\bar{X} - \mu}{\sigma / \sqrt{n}} \sim N(0, 1)$$

用 Z 来确定拒绝域，对于双侧检验，拒绝域为 $|Z| \geq Z_{\alpha/2}$ ，当 Z 的绝对值落入拒绝域时，认为原假设不成立；对于单侧检验，其拒绝域为 $Z \geq Z_{\alpha}$ 或 $Z < -Z_{\alpha}$ 。这种方法称为 Z 检验，即检验是否服从正态分布。

我们知道，R 自带的函数中只提供了 t 检验的函数 `t.test()`，而没有 Z 检验的函数，因此仿照 `t.test()`，我们可以自己编写函数 `z.test()`，用于计算 Z 统计量的值以及 P 值。

```
z.test=function(x,mu,sigma,alternative="two.sided"){
  n=length(x)
  result=list()          #构造一个空的 list，用于存放输出结果
  mean=mean(x)
  z=(mean-mu)/(sigma/sqrt(n))      #计算 z 统计量的值
  options(digits=4)          #结果显示至小数点后 4 位
  result$mean=mean;result$z=z      #将均值、z 值存入结果
  result$P=2*pnorm(abs(z),lower.tail=FALSE)  #根据 z 计算 P 值
  #若是单侧检验，重新计算 P 值
  if(alternative=="greater") result$P=pnorm(z,lower.tail=FALSE)
  else if(alternative=="less") result$P=pnorm(z)
  result
}
```

函数 `z.test()` 的输入参数为样本 x 、原假设中 μ_0 的值、已知的标准差（注意不是方差），以及检验的方向 `alternative`：默认为双侧检验“two.sided”，若是右侧检验，输入参数值为“greater”，左侧检验为“less”。

第6章介绍过程序包 BSDA (Basic Statistics and Data Analysis)，我们知道其中已经提供了函数 `z.test()`，它可以对基于正态分布的单样本和双样本进行假设检验，其使用方法如下：

```
z.test(x, y = NULL, alternative = "two.sided", mu = 0, sigma.x = NULL,
       sigma.y = NULL, conf.level = 0.95)
```

其中，`x` 和 `y` 为数值向量，默认 `y=NULL`，即进行单样本的假设检验；`alternative` 用于指定所求置信区间的类型，默认为 `two.sided`，表示求双尾的置信区间，若为 `less` 则求置信上限，`greater` 求置信下限；`mu` 表示均值，仅在假设检验中起作用，默认为 0；`sigma.x` 和 `sigma.y` 分别指定两个样本总体的标准差。

例如，在东方财富数据中心可以获得 2012 年各月北京市的新建住宅价格指数，要检验指数是否服从均值为 102.4、方差为 0.45（标准差为 0.67）的正态分布，样本数据如表 7.4 所示。

表 7.4 2012 年北京市新建住宅价格指数

1 月	2 月	3 月	4 月	5 月	6 月	7 月	8 月	9 月	10 月	11 月	12 月
102.5	102.4	102.0	101.8	101.8	102.1	102.3	102.5	102.6	102.8	103.4	104.2

原假设为样本均值等于 102.4，使用已经编写好的函数 `z.test()`，进行均值的 Z 检验：

```
> bj=c(102.5,102.4,102.0,101.8,101.8,102.1,102.3,102.5,102.6,102.8,103.4,104.2)
> z.test(x=bj,mu=102.4,sigma=0.67,alternative="two.sided")
$mean
[1] 102.5

$z
[1] 0.6894

$P
[1] 0.4906
```

使用程序包 BSDA 中的函数 `z.test()`，我们也可以得到完全相同的结果：

```
> library(BSDA)
> z.test(x=bj,mu=102.4,sigma.x=0.67,alternative="two.sided")
```

One-sample z-Test

```
data:  bj
z = 0.6894, p-value = 0.4906
alternative hypothesis: true mean is not equal to 102.4
95 percent confidence interval:
 102.1543 102.9124
sample estimates:
mean of x
 102.5333
```

检验的结果是, 由于 $P = 0.4906 > \alpha = 0.05$, 因此在 0.05 的显著性水平下, 不能拒绝原假设, 认为 2012 年各月北京的新建住宅价格指数服从均值为 102.4 的正态分布。

(2) σ^2 未知

在实际问题中, 正态总体的方差通常是未知的, 因而在 $H_0: \mu = \mu_0$ 下, 对样本均值应该进行 t 检验, 统计量为

$$T = \frac{\bar{X} - \mu_0}{S/\sqrt{n}} \sim t(n-1)$$

因此 t 检验的拒绝域为 $|T| \geq t_{\alpha/2}(n-1)$ 。若是单侧检验, 其拒绝域为 $T \geq t_{\alpha}(n-1)$ 或 $T \leq -t_{\alpha}(n-1)$ 。

方差未知的情形在 R 中实现起来比较简单, 直接调用 t 检验函数 `t.test()` 即可。

```
t.test(x, y = NULL, alternative = c("two.sided", "less", "greater"),
      mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95, ...)
```

其中, `x` 为样本数据, 若仅出现 `x`, 则进行单样本 t 检验; 若 `x` 和 `y` 同时输入, 则做双样本 t 检验; `alternative` 用于指定所求置信区间的类型, 默认为 `two.sided`, 表示求双尾的置信区间, 若为 `less` 则求置信上限, `greater` 求置信下限; `mu` 表示均值, 表示原假设中事先判断的均值, 默认值为 0; `paired` 是逻辑值, 表示是否进行配对样本 t 检验, 默认为不配对; `var.equal` 也是逻辑值, 表示双样本检验时两个总体的方差是否相等; 另外, 这个函数还可以直接计算置信区间, `conf.level` 用来表示区间的置信水平。

前面提到过, 均值的区间估计与假设检验有着紧密的内在联系, 它们本质上是对一个问题从不同角度进行讨论, 这也解释了为什么函数 `t.test()` 既可以做区间估计, 也可进行假设检验。

若上例中, 我们事先并不知道新建住宅价格指数的总体方差, 就需要用 t 检验来判断样本是否服从均值为 102.4 的正态分布。

```
> t.test(x=bj,mu=102.4)

One Sample t-test

data:  bj
t = 0.6701, df = 11, p-value = 0.5166
alternative hypothesis: true mean is not equal to 102.4
95 percent confidence interval:
 102.1 103.0
sample estimates:
mean of x
 102.5
```

检验的结果表示： t 统计量的值为 0.6701，由于 $P=0.5166>\alpha=0.05$ ，因此在 0.05 的显著性水平下，不能拒绝原假设，认为 2012 年各月北京的新建住宅价格指数服从均值为 102.4 的正态分布。

7.2.2 方差 σ^2 的检验

首先，写出检验的原假设和备择假设：

双侧检验： $H_0: \sigma^2 = \sigma_0^2, H_1: \sigma^2 \neq \sigma_0^2$

单侧检验： $H_0: \sigma^2 \leq \sigma_0^2, H_1: \sigma^2 > \sigma_0^2$ 或 $H_0: \sigma^2 \geq \sigma_0^2, H_1: \sigma^2 < \sigma_0^2$

(1) μ 已知

当均值已知时，在 $H_0: \sigma^2 = \sigma_0^2$ 下，根据 σ^2 的极大似然估计 $\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (X_i - \mu)^2$ ，有

$$\chi^2 = \frac{n\hat{\sigma}^2}{\sigma^2} \sim \chi^2(n)$$

这就转化为卡方检验的问题，用统计量 χ^2 来确定拒绝域，双侧检验的拒绝域为 $\chi^2 \geq \chi_{\alpha/2}^2(n)$ 或 $\chi^2 \leq \chi_{1-\alpha/2}^2(n)$ ；左侧检验的拒绝域为 $\chi^2 \geq \chi_{\alpha}^2(n)$ ；右侧检验的拒绝域为 $\chi^2 \leq \chi_{1-\alpha}^2(n)$ 。

(2) μ 未知

实际中更为常见的情况是总体均值未知的检验， σ^2 的极大似然估计为样本方差 S^2 。当原假设为真时，有

$$\chi^2 = \frac{(n-1)S^2}{\sigma_0^2} \sim \chi^2(n-1)$$

可见，总体均值未知与均值已知的唯一不同是，卡方检验的自由度变为 $n-1$ ，从而双侧检验的拒绝域为 $\chi^2 \geq \chi_{\alpha/2}^2(n-1)$ 或 $\chi^2 \leq \chi_{1-\alpha/2}^2(n-1)$ ；左侧检验的拒绝域为 $\chi^2 \geq \chi_{\alpha}^2(n-1)$ ；右侧检验的拒绝域为 $\chi^2 \leq \chi_{1-\alpha}^2(n-1)$ 。

R 中没有直接的函数可以做样本方差的卡方检验（只有检验卡方分布的函数），所以我们把上述两种情形写在同一个函数 `chisq.var.test()` 中，调用它就可以直接做各种情形的单样本方差检验。

```
chisq.var.test=function(x,var,mu=Inf,alternative="two.sided"){
  n=length(x)
  df=n-1 #均值未知时的自由度
  v=var(x) #均值未知时的方差估计值
  #总体均值已知的情况
  if(mu<Inf){df=n;v=sum((x-mu)^2)/n}
```

```

chi2=df*v/var #卡方统计量
options(digits=4)
result=list() #产生存放结果的列表
result$df=df;result$var=v;result$chi2=chi2;
result$P=2*min(pchisq(chi2,df),pchisq(chi2,df,lower.tail=F))
#若是单侧检验,重新计算P值
if(alternative=="greater") result$P=pchisq(chi2,df,lower.tail=F)
else if(alternative=="less") result$P=pchisq(chi2,df)
result
}

```

方差检验函数 `chisq.var.test()` 的输入参数为：样本 `x`；原假设的方差值 `var`；总体均值 `mu`，设置的默认值为 `Inf`，表示均值未知的情况，若均值已知，输入时改变参数值即可；`alternative` 表示单侧检验还是双侧检验。将此函数应用到 2012 年北京市新建住宅价格指数的案例中，如果样本方差保持在一定范围内，则说明房价比较稳定，因此我们在 0.05 的显著性水平下检验总体方差是否不超过 0.25。

$$H_0: \sigma^2 \geq 0.25, \quad H_1: \sigma^2 < 0.25$$

```

> chisq.var.test(bj,0.25,alternative="less")
$df
[1] 11
$var
[1] 0.4752
$chi2
[1] 20.91
$P
[1] 0.9656

```

检验的结果为 P 值非常大，远大于 $\alpha=0.05$ ，因此不能拒绝原假设，说明新建住宅价格指数的方差大于 0.25，变动很大。

7.3 两正态总体的检验

单正态总体的假设检验需要事先提出一个合理的参数值作为原假设，但是实际工作中很难找到一个具有实际意义的值，所以我们常常选取两个样本，就像做生物实验一样，一个作为实验组，另一个作为对照组。例如，一所学校的重点班和普通班学生的平均成绩是否有显著的差异，就需要检验两个总体的情况。

设 X 与 Y 是两个独立的总体， $X \sim N(\mu_1, \sigma_1^2)$, $Y \sim N(\mu_2, \sigma_2^2)$ ， X_1, X_2, \dots, X_{n_1} 是来自总体 X 的样本， Y_1, Y_2, \dots, Y_{n_2} 是来自总体 Y 的样本，它们的均值分别为 \bar{X} 和 \bar{Y} ，方差分别为 S_1^2 和 S_2^2 。

两正态总体的假设检验同样需要分情况讨论，即在不同检验条件下选取不同的检验方法，首

先我们将这些方法在表 7.5 中做一总结，然后再逐一展开介绍。

表 7.5 单正态总体的假设检验方法

检验对象	条件	假设检验方法	函数
比较两总体均值	方差 σ_1^2 、 σ_2^2 已知	Z 检验	z.test() (程序包 BDSA)
比较两总体均值	方差 σ_1^2 、 σ_2^2 未知但相等	t 检验	t.test() (R 自带函数)
比较两总体均值	方差 σ_1^2 、 σ_2^2 未知且不等	近似 t 检验	
成对数据的两样本均值检验	——	t 检验	
比较两总体方差	——	F 检验	var.test()
比率	精确检验	二项分布检验	binom.test()
比率	近似检验 (样本量较小)	正态检验	prop.test()

7.3.1 均值差 $\mu_1 - \mu_2$ 的检验

首先，提出三种形式下的原假设和备择假设

双侧检验： $H_0: \mu_1 - \mu_2 = 0$, $H_1: \mu_1 - \mu_2 \neq 0$

单侧检验： $H_0: \mu_1 - \mu_2 \geq 0$, $H_1: \mu_1 - \mu_2 < 0$ 或 $H_0: \mu_1 - \mu_2 \leq 0$, $H_1: \mu_1 - \mu_2 > 0$

与区间估计一样，分三种情况讨论。

(1) 两个总体的方差 σ_1^2 、 σ_2^2 已知

当两个总体方差已知时，在原假设 $\mu_1 = \mu_2$ 下，可以构造服从正态分布的统计量

$$Z = \frac{\bar{X} - \bar{Y}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \sim N(0,1)$$

因此，双侧检验的拒绝域为 $|Z| \geq Z_{\alpha/2}$ ；单侧检验的拒绝域为 $Z \geq Z_\alpha$ 或 $Z < -Z_\alpha$ 。我们可以自己编写均值差的正态检验函数 z.test2()，其与单个总体均值检验的函数非常类似。

```
z.test2=function(x,y,sigma1,sigma2,alternative="two.sided"){
  n1=length(x);n2=length(y)
  result=list()      #构造一个空的 list，用于存放输出结果
  mean=mean(x)-mean(y)
  z=mean/sqrt(sigma1^2/n1+sigma2^2/n2)    #计算 z 统计量的值
  options(digits=4)      #结果显示至小数点后 4 位
  result$mean=mean;result$z=z      #将均值、z 值存入结果
```

```

result$P=2*pnorm(abs(z),lower.tail=FALSE)    #根据 z 计算 P 值
#若是单侧检验, 重新计算 P 值
if(alternative=="greater") result$P=pnorm(z)
else if(alternative=="less") result$P=pnorm(z,lower.tail=FALSE)
result
}

```

输入参数为来自两个总体的样本 x 和 y , 以及两个总体的标准差; `alternative` 指定检验类型, 默认为 `two.sided`, 表示双侧检验 ($H_1: \mu_1 - \mu_2 \neq 0$), `greater` 表示右侧检验 ($H_1: \mu_1 - \mu_2 > 0$), `less` 表示左侧检验 ($H_1: \mu_1 - \mu_2 < 0$)。

7.2.1 节介绍的程序包 BDSA 中的函数 `z.test()` 可以快速地实现方差已知时两总体均值差的假设检验, 参数设置如下:

```

z.test(x, y = NULL, alternative = "two.sided", mu = 0, sigma.x = NULL,
       sigma.y = NULL, conf.level = 0.95)

```

首先通过参数 x 和 y 指定两个待检验样本; `alternative` 指定检验类型, 用法与 `z.test2()` 相同; `mu` 指定原假设中均值差的值, 默认为 0; `sigma.x` 和 `sigma.y` 分别指定已知的两个样本总体的标准差。

以 6.3.1 节 Bamberger's 百货公司的数据为例, 公司实施延长营业时间的改革计划, 假设已知改革前后销售额的总体标准差分别为 8 和 12, 检验这项措施对销售业绩是否有显著影响。我们希望能够得到延长营业时间后销售额更高的结论, 因此原假设和备择假设为

$$H_0: \mu_1 - \mu_2 \geq 0, \quad H_1: \mu_1 - \mu_2 < 0$$

这时应该进行左侧检验, 分别用我们自己编写的函数和程序包 BDSA 中的函数 `z.test()` 来实现这一检验。

```

> sales=read.table("d:/data/sales.txt",header=T)
> attach(sales)
> z.test2(prior,post,8,12,alternative="less")
$mean
[1] -24.54
$z
[1] -8.843
$P
[1] 4.678e-19

```

使用函数 `z.test()` 可以得到相同的结果, 同时还可以输出置信区间估计。

```

> z.test(prior,post,sigma.x=8,sigma.y=12,alternative="less")

```

```

Two-sample z-Test

```

```

data: prior and post
z = -8.8425, p-value < 2.2e-16

```

```

alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      NA -19.97758
sample estimates:
mean of x      mean of y
 60.61222    85.15519

```

结果两总体均值差的 Z 检验统计量为 $Z=-8.843$ ，对应的 P 值非常小，因此在 0.05 的显著性水平下应当拒绝原假设，表明延长营业时间后销售额更高。

(2) 两个总体的方差 σ_1^2 、 σ_2^2 未知但相等

上一章介绍过，当两个正态总体的方差已知且相等，原假设为真时可以构造服从 t 分布的统计量

$$T = \frac{(\bar{X} - \bar{Y}) - (\mu_1 - \mu_2)}{\sqrt{\left(\frac{1}{n_1} + \frac{1}{n_2}\right) S^2}} \sim t(n_1 + n_2 - 2)$$

$$\text{其中, } S^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}$$

双侧检验的拒绝域为 $|T| \geq t_{\alpha/2}(n_1 + n_2 - 2)$ ；单侧检验的拒绝域为 $T \geq t_{\alpha}(n_1 + n_2 - 2)$ 或 $T \leq -t_{\alpha}(n_1 + n_2 - 2)$ 。

(3) 两个总体的方差 σ_1^2 、 σ_2^2 未知且不等

当两总体的方差未知并且不等时，根据样本方差，可以近似得到服从 t 分布的统计量

$$T = \frac{(\bar{X} - \bar{Y}) - (\mu_1 - \mu_2)}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}} \sim t(\nu)$$

其中， ν 的估计值是

$$\hat{\nu} = \left(\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2} \right)^2 \left/ \left(\frac{(S_1^2 / n_1)^2}{n_1 - 1} + \frac{(S_2^2 / n_2)^2}{n_2 - 1} \right) \right.$$

双侧检验的拒绝域为 $|T| \geq t_{\alpha/2}(\hat{\nu})$ ；单侧检验的拒绝域为 $T \geq t_{\alpha}(\hat{\nu})$ 或 $T \leq -t_{\alpha}(\hat{\nu})$ 。

总体方差未知的 t 检验在 R 中可以直接利用 `t.test()` 完成对假设的检验。在 Bamberger's 公司的

案例中，如果改革前后总体的方差未知并且不相等，我们就可以通过 `t.test()` 直接做假设检验了，将参数 `var.equal` 设为 `F`。

```
> t.test(prior,post,var.equal=FALSE,alternative="less")

Welch Two Sample t-test

data: prior and post
t = -8.3792, df = 43.567, p-value = 6.257e-11
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
 -Inf -19.62045
sample estimates:
mean of x mean of y
 60.61222  85.15519
```

结果显示 $P = 6.257e-11 < \alpha = 0.05$ ，因此在 0.05 的显著性水平下应当拒绝原假设，表明延长营业时间后销售额更高。

7.3.2 成对数据的 t 检验

两样本的均值检验还有一种特殊情况：数据是成对出现的，即 $(X_i, Y_i), (i=1, 2, \dots, n)$ ，两个样本的容量相等。就像实验中实验组与对照组除数据外，其他条件需要保持一致，才能够得到成对的数据。

设 X_1, X_2, \dots, X_n 是来自总体 X 的样本， Y_1, Y_2, \dots, Y_n 是来自总体 Y 的样本，令 $Z_i = X_i - Y_i$ ， $(i=1, 2, \dots, n)$ ，记 $\mu = \mu_1 - \mu_2, \sigma^2 = \sigma_1^2 + \sigma_2^2$ ，则 Z_1, Z_2, \dots, Z_n 服从正态总体 $Z \sim N(\mu, \sigma^2)$ ，因而成对数据的检验其实就相当于对 Z 作单样本均值检验，可以构造统计量

$$T = \frac{\bar{Z} - \mu_0}{S/\sqrt{n}} \sim t(n-1)$$

在 R 中，利用函数 `t.test()` 就可以直接进行成对数据的 t 检验，只需要增加参数 `paired=TRUE` 即可。配对检验主要应用于生物统计和医学统计领域，下面通过例子来具体说明。

为研究某药物对高血压病人的治疗作用，用 20 名患者分组配对，分别为安慰剂组和药物组，测得血压值如表 7.6 所示。

表 7.6 某药物对患者血压的影响数据

安慰剂组	117	127	141	107	110	114	115	138	127	122
药物组	113	108	120	107	104	98	102	132	120	114

为测定该药物是否有降压作用，原假设为药物无作用，即安慰剂组的均值小于药物组，采用

右侧检验。在 R 中输入以下代码：

```
> x=c(117,127,141,107,110,114,115,138,127,122)
> y=c(113,108,120,107,104,98,102,132,120,114)
> t.test(x,y,paired=TRUE,alternative="greater")

Paired t-test

data: x and y
t = 4.5856, df = 9, p-value = 0.0006586
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 6.002487      Inf
sample estimates:
mean of the differences
      10
```

检验的结果显示， P 值远小于 $\alpha = 0.05$ ，因此拒绝原假设，说明药物组的均值明显降低，该药物有降压作用。

此外，程序包 DAAG 中的函数 onesamp() 专门用于成对样本的 t 检验，其调用格式如下

```
onesamp(dset, x="unsprayed", y="sprayed", xlab=NULL, ylab=NULL, dubious=NULL,
        conv=NULL, dig=2)
```

其中，dset 是一个具有两列的矩阵或数据框；x 是 dset 的其中一列，发挥“预测”作用；y 是另一列，发挥“响应”作用；dubious 是逻辑值 (FALSE/TRUE) 向量，以省略指定的点；conv 指定缩放系数，并应用于数据。

7.3.3 两总体方差的检验

假设两个样本分别来自总体 X 、 Y 且独立，通过检验比较它们的方差，首先提出假设

双侧检验： $H_0: \sigma_1^2 = \sigma_2^2, H_1: \sigma_1^2 \neq \sigma_2^2$

单侧检验： $H_0: \sigma_1^2 \leq \sigma_2^2, H_1: \sigma_1^2 > \sigma_2^2$ 或 $H_0: \sigma_1^2 \geq \sigma_2^2, H_1: \sigma_1^2 < \sigma_2^2$

实际问题中的总体均值通常是未知的，那么当 H_0 为真时，有

$$F = \frac{S_1^2}{S_2^2} \sim F(n_1 - 1, n_2 - 1)$$

通过 F 分布来确定拒绝域，双侧检验的拒绝域为 $F \geq F_{\alpha/2}(n_1 - 1, n_2 - 1)$ 或 $F \leq F_{1-\alpha/2}(n_1 - 1, n_2 - 1)$ ；右侧检验的拒绝域为 $F \geq F_{\alpha}(n_1 - 1, n_2 - 1)$ ；左侧检验为 $F \leq F_{1-\alpha}(n_1 - 1, n_2 - 1)$ 。

R 中的函数 var.test() 可以做方差比较的 F 检验以及相应的区间估计，其调用格式在 6.3.2 节中已经介绍过。仍以 Bamberger's 公司的数据为例，如果想要考察延长营业时间前后周营业额的波

动情况是否一致，可以通过 F 检验完成。

```
> var.test(prior,post)

      F test to compare two variances

data:  prior and post
F = 0.3889, num df = 26, denom df = 26, p-value = 0.01914
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.1772458 0.8534348
sample estimates:
ratio of variances
 0.3889315
```

检验结果的 $P = 0.01914 < \alpha = 0.05$ ，故拒绝原假设，说明延长营业时间前后销售额的方差不相同。

7.4 比率的检验

总体比率是研究在一个总体中具有某种特征的个体所占比值的问题，从概念上就可以发现，它与二项分布模型有着密切的联系。假设 X_1, X_2, \dots, X_n 分别是来自伯努利分布（两点分布）的独立样本， $X_i \sim \text{binom}(1, p)$ ，那么根据分布性质可以得到 $T = \sum_{i=1}^n X_i \sim \text{binom}(n, p)$ ，即样本取值之和服从二项分布，其中的参数 p 即我们所要检验的比率。首先提出假设

双侧检验： $H_0: p = p_0, H_1: p \neq p_0$

单侧检验： $H_0: p \leq p_0, H_1: p > p_0$ 或 $H_0: p \geq p_0, H_1: p < p_0$

7.4.1 比率的二项分布检验

根据上面的理论，我们可以对比率 p 的取值作比较精确的检验，即检验统计量 T 是否服从参数为 p 的二项分布。在 R 中使用函数 `binom.test()` 完成：

```
binom.test(x, n, p = 0.5, alternative = c("two.sided", "less", "greater"), conf.level = 0.95)
```

其中， n 是样本总数， x 是样本具有某种特征的个体数，也可以将这两个值都存放在二维向量 x 中，这时 n 忽略； p 值为原假设中的概率值。使用 6.4 节中的案例，2000 户家庭中人均不足 5 平米的困难户有 214 个，政府希望将总体中困难户的比率控制在 10% 左右，可通过假设检验来判断这一目标是否达到。

```
> binom.test(214, 2000, p=0.1)

Exact binomial test
```

```

data: 214 and 2000
number of successes = 214, number of trials = 2000, p-value = 0.2966
alternative hypothesis: true probability of success is not equal to 0.1
95 percent confidence interval:
 0.09378632 0.12137786
sample estimates:
probability of success
0.107

```

由于 $P = 0.2966 > \alpha = 0.05$, 故不能拒绝原假设, 说明总体居民的困难户比率保持在 10% 左右。除了 P 值, 检验结果还给出了置信区间和样本比率估计值 0.107。

7.4.2 比率的近似检验

二项分布检验是比率的精确检验, 适用于样本量较小的情况。但上一章提到过, 当样本量足够大时, 根据中心极限定理可知样本比率的抽样分布近似服从正态分布, 从而可转化为正态分布检验的问题。因为在原假设 $p = p_0$ 下, 我们可以构造出统计量

$$Z = \frac{\hat{p} - p_0}{\sqrt{p_0(1-p_0)/n}} \sim N(0,1)$$

R 语言中实现比率的近似正态检验的函数是 `prop.test()`, 6.4 节介绍了其调用格式。

大样本通常指样本容量大于 30, 在上面的案例中, 样本量为 2000, 因此可以使用正态检验方法代替二项分布。

```

> prop.test(214,2000,p=0.1)

1-sample proportions test with continuity correction

data: 214 out of 2000, null probability 0.1
X-squared = 1.0125, df = 1, p-value = 0.3143
alternative hypothesis: true p is not equal to 0.1
95 percent confidence interval:
 0.09396256 0.12157198
sample estimates:
p
0.107

```

正态检验的 P 值 $= 0.3143 > \alpha = 0.05$, 故不能拒绝原假设, 说明总体居民的困难户比率保持在 10% 左右。

7.5 非参数的检验

上面介绍的在总体分布形式已知的情况下, 对总体分布的参数如均值、方差等进行推断的方

法统称为参数检验。但在实际的数据分析过程中，由于种种原因，人们往往无法对总体分布形态作简单假定，此时参数检验的方法就不再适用了。基于这种考虑，非参数检验可以在总体方差未知或知道甚少的情况下，利用样本数据对总体分布形态等进行推断。非参数检验是统计分析方法的重要组成部分，它与参数检验共同构成统计推断的基本内容。由于非参数检验方法在推断过程中不涉及有关总体分布的参数，因而得名为“非参数”检验。

非参数检验的优点可以总结为 3 点：

1. 检验条件宽松，适用范围宽。它适合处理如非正态的、方差不等的或者分布形状未知的数据检验。
2. 检验方法灵活，用途广泛。它不但可以处理测量层次较高的定距、定比数据，也可以处理层次较低的定类、定序数据。
3. 检验计算简单。由于自由分布的检验方法不用复杂计算，因此非参数检验比较直观。

统计中的非参数统计方法有很多，涉及范围较广，这里我们简要介绍几种在 R 软件中应用较多的非参数检验。

7.5.1 总体分布的 χ^2 检验

在第 5 章，我们介绍了用直方图、QQ 图和经验分布图等方式来描述观测数据是否服从某种分布，更进一步的方法，可以通过参数检验来完成分布的拟合优度检验，如前几章介绍过的 W 正态检验、K-S 检验等。参数检验认为样本具有某种指定的分布形式，只是其中的一些参数未知，因此要检验的假设是某个参数落在特定范围内。接下来我们要介绍如何使用非参数统计方法检验观测数据是否服从某种分布，其假设不再是针对具体参数，而是针对分布类型，需要分两种情况讨论。

(1) 理论分布已知

已知随机变量 X 应服从分布 F ，现得到 X 的一个样本数据 X_1, X_2, \dots, X_n ，对它做非参数卡方检验，首先确定原假设为

$$H_0: X \text{ 具有分布 } F$$

因此，备择假设为

$$H_1: X \text{ 不具有分布 } F$$

对该假设的检验方法如下。首先将数轴 $(-\infty, \infty)$ 分成 m 个区间：

$$I_1 = (-\infty, a_1), I_2 = [a_1, a_2), \dots, I_m = [a_{m-1}, \infty)$$

记样本落在这些区间的理论概率分别为

$$p_1, p_2, \dots, p_m, \quad p_i = P\{X \in I_i\}, i = 1, 2, \dots, m$$

那么在原假设下, 样本 X_1, X_2, \dots, X_n 落入区间 I_i 内的个数的期望值为 np_i , 记 n_i 分别为样本落入区间 I_i 的实际个数, 因此 n_i 和 np_i 的差距即可视为理论值与观测值之间的偏差, 由此得到 Pearson χ^2 统计量

$$K = \sum_{i=1}^m \frac{(n_i - np_i)^2}{np_i}$$

Pearson 证明在原假设成立的条件下, 当 $n \rightarrow \infty$ 时, K 依分布收敛于自由度为 $m-1$ 的 χ^2 分布。基于这一原理, 可以引进一个大样本检验: 给定显著性水平 α , 若 $K > \chi_\alpha^2(m-1)$, 则拒绝原假设。这就是 Neyman-Pearson 拟合优度卡方检验。

根据 K 值, 我们可以进一步得到检验的 P 值, 比较 P 值和显著性水平 α 来判断是否接受原假设会更加容易。

$$P = P\{\chi_\alpha^2(m-1) < K\}$$

P 值越大, 越支持原假设为真; 反之, 当 P 值 $< \alpha$ 时, 就应拒绝原假设。

R 软件中提供了实现 Pearson 拟合优度卡方检验的函数 `chisq.test()`, 其调用格式为

```
chisq.test(x, y = NULL, correct = TRUE, p = rep(1/length(x), length(x)),
           rescale.p = FALSE, simulate.p.value = FALSE, B = 2000)
```

其中, x 是样本数据的向量或矩阵; y 是与 x 长度相同的向量, 当 x 是矩阵时忽略参数 y ; `correct` 是逻辑变量, 说明计算检验统计量时是否进行连续修正, 默认为 `TRUE`, 即修正; p 是原假设落在区间 I_i 内的理论概率, 默认值表示假设样本服从均匀分布; `rescale.p` 也是逻辑变量, 默认值为

`FALSE`, 要求输入的概率值满足 $\sum_{i=1}^m p_i = 1$, 若不满足将报错, 若 `rescale.p=FALSE` 则不要求这一点, 当 p 值之和不等于 1 时, 程序将重新计算 p ; `simulate.p.value` 是逻辑值 (默认为 `FALSE`), 当它为 `TRUE` 时, 将采用仿真方法计算 p 值, 这时, 参数 B 指定仿真的次数。

下面通过案例来说明非参数卡方检验的实际应用。在 7.2.1 节中, 提到了 2012 年北京市新建住宅价格指数的案例 (数据如表 7.7 所示), 接下来用 Pearson χ^2 检验来判断北京新建住宅价格指数是否服从正态分布。

表 7.7 2012 年北京市新建住宅价格指数

1 月	2 月	3 月	4 月	5 月	6 月	7 月	8 月	9 月	10 月	11 月	12 月
102.5	102.4	102.0	101.8	101.8	102.1	102.3	102.5	102.6	102.8	103.4	104.2

根据上面的理论，非参数卡方检验应分 4 个步骤进行，分别编写 R 语句来实现。

第一步，直接输入数据向量 `bj`，同时可以绘制直方图，帮助我们对数据的分布形态有初步了解。

```
> bj=c(102.5,102.4,102.0,101.8,101.8,102.1,102.3,102.5,102.6,102.8,103.4,104.2)
> hist(bj)
```

绘制结果如图 7.3 所示。

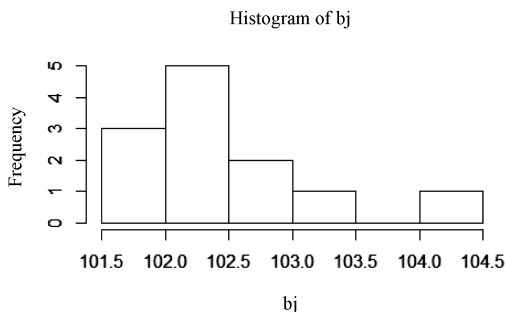


图 7.3 北京市新建住宅价格指数直方图

第二步，对数据进行分组，计算各组的频数，从而得到样本落入各区间的实际个数 n_i 。分组主要依靠观察图形后的主观判断，例如我们把数据分成 4 组：其中 $A_1 = \{101.5 \leq X < 102.0\}$, $A_2 = \{102.0 \leq X < 102.5\}$, $A_3 = \{102.5 \leq X < 103.0\}$, $A_4 = \{103.0 \leq X < 104.5\}$ 。在 R 中实现这一过程需要用到函数 `table()` 和 `cut()`，下面简要介绍这两个函数的用法。

首先函数 `cut()` 用于将变量的区域分成若干区间，其调用格式为

```
cut(x, breaks, labels = NULL, include.lowest = FALSE,
    right = TRUE, dig.lab = 3, ordered_result = FALSE, ...)
```

其中，`x` 是数据向量；`breaks` 是表示区间端点的数值向量，可简写为 `br`。

而函数 `table()` 可以计算因子合并后的个数，以列联表的形式展示出每个区间的数据频数。

```
table(..., exclude = if (useNA == "no") c(NA, NaN), useNA = c("no", "ifany", "always"),
    dnn = list.names(...), deparse.level = 1)
```

其中参数 `exclude` 指定需要剔除的因子水平，如果设置为 `NULL`，则意味着参数 `useNA = "always"`。需要注意的一点是，`table` 合并的区间形式为“左开右闭”，所以下面设置 `cut()` 中的参数 `breaks` 时要做一些调整。

```
> A=table(cut(bj,breaks=c(101.4,101.9,102.4,102.9,104.5))) #两个函数嵌套使用
#注意端点 breaks 的写法
> A
(101.4,101.9] (101.9,102.4] (102.4,102.9] (102.9,104.5]
           2           4           4           2
```

第三步, 计算原假设条件下数据落入各区间的理论值, 由于本例中原假设是正态分布, 所以先计算样本均值和标准差, 再用正态分布函数 `pnorm()` 计算落入各小区间内的理论概率 p_i 。

```
> br=c(101.5,102,102.5,103,104.5)
> p=pnorm(br,mean(bj),sd(bj)) #注意 pnorm() 计算出的是分布函数
> p=c(p[1],p[2]-p[1],p[3]-p[2],1-p[3])
> options(digits=2)
> p
[1] 0.067 0.153 0.261 0.519
```

第四步, 利用函数 `chisq.test()` 完成 Pearson 拟合优度卡方检验。

```
> chisq.test(A,p=p)
Chi-squared test for given probabilities

data: A
X-squared = 7.5, df = 3, p-value = 0.05849
```

总体分布的卡方检验结果为 $P=0.05849 > \alpha=0.05$, 因此在 0.05 的显著性水平下, 不能够拒绝原假设, 可以认为北京市新建住宅价格指数服从正态分布。

(2) 理论分布依赖于 r 个未知参数

若总体的分布 F 依赖于 r 个未知参数 $\theta_1, \theta_2, \dots, \theta_r$, 对 X 的样本数据 X_1, X_2, \dots, X_n 进行非参数卡方检验, 这时原假设为

$$H_0: X \text{ 具有分布 } F(x, \theta_1, \theta_2, \dots, \theta_r)$$

要实现这一检验, 首先需要根据样本计算 $(\theta_1, \theta_2, \dots, \theta_r)$ 的极大似然估计 $(\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_r)$, 这时原假设为

$$H_0: X \text{ 具有分布 } F(x, \hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_r)$$

之后按照理论分布已知的情况处理即可, 不同在于 K 统计量服从自由度为 $m-r-1$ 的 χ^2 分布。

7.5.2 Kolmogorov-Smirnov 检验

在统计学中, Kolmogorov-Smirnov (KS) 检验基于累计分布函数, 用以检验变量是否符合某种分布或比较两组之间有无显著性差异。KS 检验有单样本和双样本检验之分。

(1) 单样本 KS 检验

Kolmogorov-Smirnov 检验是用来检验一个数据的观测经验分布是否是已知的理论分布，当两者之间的差距很小时可以认为该样本取自已知的理论分布。KS 检验通过经验分布与假设分布的上确界来构造统计量，因此它可以检验任何分布类型，原假设为

$$H_0: X \text{ 具有分布 } F$$

检验的统计量为

$$Z = \sqrt{n} \max_i (|F_n(x_{i-1}) - F(x_i)|, |F_n(x_i) - F(x_i)|)$$

在 R 中用函数 `ks.test()` 可以直接得到 KS 检验的结果， P 值越大，越支持原假设为真；反之，当 P 值 $< \alpha$ 时，就应拒绝原假设。R 中自带的函数 `ks.test()` 来自于程序包 `stats`，而程序包 `dgof` 中也包含 `ks.test()`，两者的使用方法几乎完全相同。该函数既可以进行单样本的 KS 检验，又可以进行双样本检验，其调用格式为

```
ks.test(x, y, ..., alternative = c("two.sided", "less", "greater"), exact = NULL)
```

其中， x 是数据向量；参数 y 非常多样化，可以是一个数据向量，或是一个字符串用于指定累积分布名称，还可以是实际的累积分布函数，例如 `pnorm`，另外， y 可以是 `ecdf` 函数对象，用来指定一个离散分布；“...” 给出 y 指定分布的参数；`alternative` 设置检验类型；`exact` 为 `NULL` 或者一个逻辑值，用于指定 P 值是否应该被计算。

对一台设备进行寿命检验，记录 10 次无故障工作时间，检验其是否服从参数为 1/1500 的指数分布，数据如下：

```
420 500 920 1380 1510 1650 1760 2100 2300 2350
```

调用函数 `ks.test()` 来完成检验：

```
> X=c(420,500,920,1380,1510,1650,1760,2100,2300,2350)
> ks.test(X,"pexp",1/1500) #pexp 为指数分布累积分布函数的名称，1/1500 为指数分布参数
One-sample Kolmogorov-Smirnov test

data: X
D = 0.3, p-value = 0.2654
alternative hypothesis: two-sided
```

单样本 KS 检验的结果为 P 值=0.2654，其大于显著性水平 0.05，因此不能拒绝原假设，说明该设备的寿命服从 $\lambda = 1/1500$ 的指数分布。

(2) 两样本 KS 检验

两样本 KS 检验由于对两样本的经验分布函数的位置和形状参数的差异都比较敏感，因此成为比较两样本的最有用且最常规的非参数方法之一。

假设有分别来自两个独立总体的两样本，要想检验它们背后的总体分布是否相同，就可以进行两独立样本的 KS 检验。原理与单样本相同，只需要把原假设中的分布换成另一个样本的经验分布即可。假定两个样本的样本量分别为 n_1 和 n_2 ，累积经验分布函数分别为 $F_1(x)$ 和 $F_2(x)$ ，它们均为连续分布函数且未知，检验这两分布是否相同，即原假设为

$$H_0: F_1(x) = F_2(x)$$

记 $D_j = F_1(x_j) - F_2(x_j)$ ，检验统计量为

$$Z = \max_j |D_j| \sqrt{\frac{n_1 n_2}{n_1 + n_2}}$$

Z 近似正态分布。

举一个双样本 KS 检验的实例，有分别从两个总体抽取的 25 个和 20 个观测值的随机样本，数据如表 7.8 所示，下面对它们进行检验，判断它们是否来自同一分布。

表 7.8 两样本 KS 检验数据

$F_1(x)$	0.61 0.29 0.06 0.59 -1.73 -0.74 0.51 -0.56 0.39 1.64 0.05 -0.06 0.64 -0.82 0.37 1.77 1.09 -1.28 2.36 1.31 1.05 -0.32 -0.40 1.06 -2.47
$F_2(x)$	2.20 1.66 1.38 0.20 0.36 0.00 0.96 1.56 0.44 1.50 -0.30 0.66 2.31 3.29 -0.27 -0.37 0.38 0.70 0.52 -0.71

在 R 中输入数据并调用 `ks.test()` 完成两样本检验。

```
>xx=c(0.61,0.29,0.06,0.59,-1.73,-0.74,0.51,-0.56,0.39,1.64,0.05,-0.06,0.64,
      -0.82,0.37,1.77,1.09,-1.28,2.36,1.31,1.05,-0.32,-0.40,1.06,-2.47)
>yy=c(2.20,1.66,1.38,0.20,0.36,0.00,0.96,1.56,0.44,1.50,-0.30,0.66,2.31,
      3.29,-0.27,-0.37,0.38,0.70,0.52,-0.71)
> ks.test(xx,yy)
```

Two-sample Kolmogorov-Smirnov test

```
data: xx and yy
D = 0.23, p-value = 0.5286
alternative hypothesis: two-sided
```

两样本 KS 检验的结果为 $P=0.5286$ ，远远大于显著性水平 0.05，因此不能拒绝原假设，可以认为两个样本来自同一分布。

(3) KS 检验与卡方检验的比较

KS 检验与卡方检验的相同之处在于它们都是采用实际频数和期望频数之差进行检验。但不

同点在于，卡方检验必须先将数据分组才能获得实际的观测频数，而 KS 检验法可以直接对原始数据的 n 个观测值进行检验，所以它对数据的利用更完整。另外在使用范围上，卡方检验主要用于分类数据，而 KS 检验主要用于有计量单位的连续和定量数据。

KS 检验作为一种非参数方法，具有稳健性。它不依赖于均值的位置，对数据量纲不敏感，一般来讲比卡方检验更有效。与其他参数检验不同，KS 检验的适用范围非常广，不像 t 检验一样局限于正态分布（当数据偏离较大时 t 检验会失效）。

第 8 章

方差分析及 R 实现

方差分析 (Analysis of Variance, ANOVA) 广泛应用于商业、经济、医学、农业等诸多领域的数量分析研究中。例如商业广告宣传方面, 广告效果可能会受广告形式、地区规模、播放时段、播放频率等多个因素的影响, 通过方差分析研究众多因素中, 哪些是主要的以及如何产生影响等。而在经济管理中, 方差分析常用于分析变量之间的关系, 如人民币汇率对股票收益率的影响、存贷款利率对债券市场的影响, 等等。

协方差是在方差分析的基础上, 综合回归分析的方法, 研究如何调节协变量对因变量的影响效应, 从而更加有效地分析实验处理效应的一种统计技术。

本章将介绍这两种分析方法的理论基础及在 R 语言中的实现。

8.1 单因素方差分析及 R 实现

方差分析主要分为两类: 单因素方差分析和双因素方差分析。当方差分析中只涉及一个分类变量时, 称为单因素方差分析, 适用于单因素 A 拥有两个以上水平时, 研究各水平对因变量变异的影响。首先了解方差分析中的一些基本术语, 我们所要检验的对象称为因素或因子; 因素的不同表现称为水平; 而每个因子水平下得到的样本数据即观测值。

8.1.1 基本假设的检验

方差分析中有 3 个基本假设:

- ① 正态假设。对于因素的每个水平, 其观测值都是来自正态总体的随机样本;
- ② 方差齐性假设。各个总体的方差相同;
- ③ 独立假设。观测值之间都是独立的。

设试验中的因素为 A ，有 r 个水平 A_1, A_2, \dots, A_r ，在每个水平下进行试验得到结果 $x_{i1}, x_{i2}, \dots, x_{in_i}, i = 1, 2, \dots, r$ ，其被看作是来自第 i 个正态总体 $X_i \sim N(\mu_i, \sigma^2)$ ，其中参数未知且每个样本都独立。从而单因素方差分析的数学模型可以表示为一种线性模型

$$\begin{cases} x_{ij} = \mu + \alpha_i + \varepsilon_{ij}; i = 1, 2, \dots, r, j = 1, 2, \dots, n_i \\ \varepsilon_{ij} \sim N(0, \sigma^2) \text{ 且相互独立} \\ \sum_{i=1}^r n_i \alpha_i = 0 \end{cases}$$

其中 μ 是所有总体的均值， $\alpha_i = \mu_i - \mu$ 称为第 i 个水平的效应， ε_{ij} 是随机误差。

在进行具体的方差分析之前，先对几条假设进行检验，由于数据是随机抽取的，我们假设总体满足独立性，但对正态性和方差齐次性往往没有把握，所以要进行这两项检验。

(1) 正态性检验

对数据的正态性，利用 Shapiro-Wilk 正态检验方法 (W 检验)，它通常用于样本容量 $n \leq 50$ 时，检验样本是否符合正态分布。 W 检验是建立在次序统计量的基础上的，将 n 个独立观测值按非降次序排列，记为 $x_{(1)}, x_{(2)}, \dots, x_{(n)}$ ，构造 W 检验统计量为

$$W = \frac{[\sum_{i=1}^n a_i (x_{n+1-i} - x_i)]^2}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

其中，系数 a_i 为样本容量为 n 时的系数。可以证明，总体分布为正态分布时， W 的值应该接近 1，通过查表可以获得 W 统计量的 α 分位数 W_α ，当 $W \leq W_\alpha$ 时，在显著性水平 α 下应拒绝原假设，说明所检验的数据不服从正态分布；当 $W > W_\alpha$ 时，无法拒绝原假设，则数据服从正态分布。

在 R 软件中，函数 `shapiro.test()` 提供了 W 统计量和相应 P 值，所以可以直接使用 P 值作为判断标准，其调用格式为 `shapiro.test(x)`，参数 x 即所要检验的数据集，它是长度在 3~5000 之间的向量。

上文提到方差分析广泛应用于经济管理领域，接下来我们举例说明，某银行规定 VIP 客户的月均账户余额要达到 100 万元，并以此作为比较各分行业绩的一项指标。这里分行即因子，账户余额是所要检验的指标，先从三个分行（对应三个水平 A_1 、 A_2 和 A_3 ）中，分别随机抽取 7 个 VIP 客户的账户，数据列在表 8.1 中。为了用单因素方差分析判断三个分行此项业绩指标是否相同，首先对三个分行的账户余额分别进行正态检验。

表 8.1 各分行样本的账户余额

分行	账户余额 (万元)						
A_1	103	101	98	110	105	100	106
A_2	113	107	108	116	114	110	115
A_3	82	92	84	86	84	90	88

```
> x1=c(103,101,98,110,105,100,106)
> x2=c(113,107,108,116,114,110,115)
> x3=c(82,92,84,86,84,90,88)
> shapiro.test(x1)
```

```
Shapiro-Wilk normality test
data: x1
W = 0.9778, p-value = 0.948
> shapiro.test(x2)
```

```
Shapiro-Wilk normality test
data: x2
W = 0.9189, p-value = 0.4607
> shapiro.test(x3)
```

```
Shapiro-Wilk normality test
data: x3
W = 0.9547, p-value = 0.7724
```

P 值均大于显著性水平 $\alpha = 0.05$ ，因此不能拒绝原假设，说明数据在因子 A 的三个水平下都是来自正态分布的。

(2) 方差齐性检验

方差分析的另一个假设——方差齐性，需要检验不同水平下的数据方差是否相等。很多统计学家提出了一些很好的方法，这里首先介绍 R 中最常用的 Bartlett 检验。原假设为

$$H_0: \sigma_1^2 = \sigma_2^2 = \cdots = \sigma_r^2, \quad H_1: \sigma_i^2 \text{ 不全相等}$$

Bartlett 检验适用于多样本的情形，在每个水平下都有

$$S_i^2 = \frac{1}{n_i - 1} \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_{i\cdot})^2$$

$$S^2 = \frac{1}{n - r} \sum_{i=1}^r (n_i - 1) S_i^2$$

$$c = 1 + \frac{1}{3(r-1)} \left[\sum_{i=1}^r \frac{1}{(n_i - 1)} - \frac{1}{n - r} \right], n = \sum_{i=1}^r n_i$$

在原假设成立时，可以得到卡方统计量

$$\chi^2 = \frac{2.3026}{c} \left[(n-r) \ln S^2 - \sum_{i=1}^r (n_i - 1) \ln S_i^2 \right] \sim \chi^2(r-1)$$

它近似服从自由度为 $r-1$ 的卡方分布，因此我们就可以根据统计量或 P 值来判断是否接受原假设。

R 中的 Bartlett 检验函数是 `bartlett.test()`，其调用格式为

```
bartlett.test(x, g, ...)
```

其中，参数 x 是数据向量或列表 (list)； g 是因子向量，如果 x 是列表则忽略 g 。当使用数据集时，也通过 `formula` 调用函数

```
bartlett.test(formula, data, subset, na.action, ...)
```

`formula` 是形如 $\text{lhs} \sim \text{rhs}$ 的方差分析公式；`data` 指明数据集；`subset` 是可选项，可以用来指定观测值的一个子集用于分析；`na.action` 表示遇到缺失值时应当采取的行为。

继续上面的例子，首先构造完整的数据集 `account`，数据框的第一个元素是账户余额数值，第二个元素是因子的水平，构造时使用函数 `factor()`，因子水平为 1、2、3，每个水平下有 7 个观测值，因此设置 `each=7`。再做 Bartlett 检验。

```
> x=c(x1,x2,x3)
> account=data.frame(x,A=factor(rep(1:3,each=7)))
> bartlett.test(x~A,data=account)

Bartlett test of homogeneity of variances
data:  x by A
Bartlett's K-squared = 0.1362, df = 2, p-value = 0.9341
```

由于 P 值远远大于显著性水平 $\alpha = 0.05$ ，因此不能拒绝原假设，我们认为不同水平下的数据是等方差的。

8.1.2 单因素方差分析

方差分析的目的是要比较因素 A 的 r 个水平下，试验结果是否有显著差异。以样本均值作为检验的标准，写出检验假设

$$H_0: \alpha_1 = \alpha_2 = \cdots = \alpha_r, \quad H_1: \alpha_1, \alpha_2, \dots, \alpha_r \text{ 不全相等}$$

如果拒绝原假设，说明样本来自不同的正态总体，则由因素 A 的各个水平所造成均值的差异有统计意义；若不能拒绝原假设，说明样本来自相同的正态总体，因素的不同水平之间无差异。

为了度量这种差异，我们将描述所有观测数据离散程度的总离差平方和分解为两部分，研究

差异的主要来源，所以叫做“方差分析”。总离差平方和（也称为总变差）由 SS_T 表示。

$$SS_T = \sum_{i=1}^r \sum_{j=1}^{n_i} (x_{ij} - \bar{x})^2, \quad \bar{x} = \frac{1}{n} \sum_{i=1}^r \sum_{j=1}^{n_i} x_{ij}$$

总离差平方和可以分解为两部分：组内平方和 SS_E ，它针对同一个正态总体的样本计算内部差异情况，体现随机误差的影响；另一部分为组间平方和 SS_A ，表示因子各水平下的样本均值与总平均值之间的差平方和，反映 r 个总体均值之间的差异，因此 SS_A 越大，说明因素 A 对试验结果的影响越大。

$$SS_T = SS_E + SS_A$$

其中

$$SS_E = \sum_{i=1}^r \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)^2, \quad \bar{x}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_{ij}$$

$$SS_A = \sum_{i=1}^r \sum_{j=1}^{n_i} (\bar{x}_i - \bar{x})^2, \quad n = n_1 + n_2 + \cdots + n_r$$

在原假设的条件下，经过统计分析可以得出 σ^2 的无偏估计 $SS_E/(n-r)$ 和 $SS_A/(r-1)$ ，所以以下式子成立

$$\frac{SS_E}{\sigma^2} \sim \chi^2(n-r), \quad \frac{SS_A}{\sigma^2} \sim \chi^2(r-1)$$

并且 SS_A 与 SS_E 相互独立，于是可以构造出 F 统计量

$$F = \frac{SS_A/(r-1)}{SS_E/(n-r)} \sim F(r-1, n-r)$$

在显著性水平 α 下， $F_\alpha(r-1, n-r)$ 是 F 分布上的 α 分位点。若 $F > F_\alpha(r-1, n-r)$ ，则拒绝原假设，表明所检验的因素对观测值有显著影响，当然也可以通过 P 值决定是否拒绝原假设，R 的分析结果中计算了 P 值，所以我们直接用它来判断。

为了使方差分析的计算过程更加清晰，我们通常将上述过程列在一张表内，即方差分析表，如表 8.2 所示。

表 8.2 因素方差分析表

方差来源	自由度	平方和	均方	F 值	P 值
因素 A	$r-1$	SS_A	$MS_A = \frac{SS_A}{r-1}$	$F = \frac{MS_A}{MS_E}$	p

续表

方差来源	自由度	平方和	均方	F值	P值
误差	$n-r$	SS_E	$MS_E = \frac{SS_E}{n-r}$		
总和	$n-1$	SS_T			

R 中的函数 `aov()` 用于方差分析的计算，其调用格式为

```
aov(formula, data = NULL, projections = FALSE, qr = TRUE, contrasts = NULL, ...)
```

其中的参数 `formula` 表示方差分析的公式，在单因素方差分析中即为 $x \sim A$ ；`data` 表示做方差分析的数据框；`projections` 为逻辑值，表示是否返回预测结果；`qr` 同样是逻辑值，表示是否返回 QR 分解结果，默认为 TRUE；`contrasts` 是公式中的一些因子的对比列表。通过函数 `summary()` 可列出方差分析表的详细结果。

上面的例子已经对数据的正态性和方差齐性做了检验，接下来就可以进行方差分析。

```
> a.aov=aov(x~A,data=account)
> summary(a.aov)
      Df Sum Sq Mean Sq F value    Pr(>F)
A       2   2315    1158   82.68 8.46e-10 ***
Residuals 18    252      14
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

输出的分析结果即对应一个方差分析表， A 即因子，`Residuals` 表示残差。各列分别为：`Df` 表示自由度，`Sum Sq` 为平方和，`Mean Sq` 为均方和，`F value` 表示 F 检验统计量的值。检验的 P 值远远小于 $\alpha = 0.05$ ，说明拒绝原假设，不同分行的经济业绩有显著差别。使用 `plot()` 函数，我们可以直观地描述这种差异。

```
> plot(account$x~account$A)
```

绘制结果如图 8.1 所示。

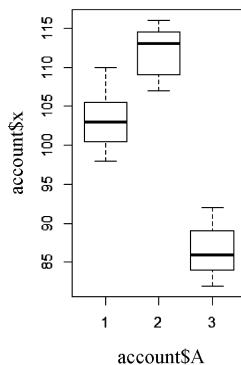


图 8.1 不同分行的经营业绩

R 中方差分析的其他实现方法:

① 单因素方差分析的数学模型可看作是一种特殊的线性模型, 因此方差分析还可以通过线性模型函数 `lm()` (下一章详细介绍) 计算得到, 再用函数 `anova()` 提取其中的方差分析表, 因此 `aov(formula)` 等价于 `anova(lm(formula))`。

```
> anova(lm(x~A,data=account))
Analysis of Variance Table

Response: x
      Df Sum Sq Mean Sq F value    Pr(>F)    
A       2  2315.1  1157.6   82.684 8.464e-10 ***
Residuals 18   252.0    14.0                
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

② 单因素方差分析还可以通过函数 `oneway.test()` 实现。

```
oneway.test(formula, data, subset, na.action, var.equal = FALSE)
```

若各水平下的总体方差相等, 且参数 `var.equal` 设置为 `TRUE`, 则等同于函数 `aov()` 所作的方差分析结果。在默认情况下, 各水平下的总体方差不相等 (`var.equal=FALSE`), 则使用 Welch (1951) 的近似方法, 在任意多个样本的情况下作两样本 Welch 检验。在满足方差齐性的条件下, 用 `oneway.test()` 做检验。

```
> oneway.test(x~A,data=account,var.equal=TRUE)

One-way analysis of means
data: x and A
F = 82.6837, num df = 2, denom df = 18, p-value = 8.464e-10
```

Levene 检验

我们知道, 在用数据作方差分析之前, 必须检验它是否满足几个基本假设。常见的 Bartlett 多样本方差齐性检验主要用于正态分布的数据, 对于非正态分布的数据, 检验效果不理想。因此我们介绍方差齐性检验的另一个方法——Levene 检验, 它既可以用于正态分布的数据, 也可用于非正态分布的数据或分布不明的数据, 具有比较稳健的特点, 检验效果也比较理想。

Levene 检验先对原始数据作变换, 然后再进行方差分析, 变换主要有三种方式:

- ① $d_{ij} = |X_{ij} - \bar{X}_i|$, \bar{X}_i 是原始数据中第 i 个水平下样本的简单平均。
- ② $d_{ij} = |X_{ij} - md_i|$, md_i 是第 i 个水平下样本的中位数。
- ③ $d_{ij} = |X_{ij} - \bar{X}_i|^2$, 作离均差平方变换。

其中第一种变换方式主要用于对称分布或正态分布的数据，第二种变换可用于偏态分布的数据。对变换后的数据作方差分析，根据 P 值判断各组原始数据的方差齐性是否满足。

R 的程序包 `car` 中提供了 Levene 检验的函数 `levene.test()`，程序包无需安装，但使用前要先加载，调用格式为

```
levene.test(x, group)
```

其中， x 是数据向量， $group$ 是因子向量。

```
> library(car)
> levene.test(account$x, account$A)
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  2  0.0426 0.9584
      18
```

由于 P 值大于 $\alpha = 0.05$ ，不能拒绝原假设，我们认为不同水平下的数据是等方差的。

8.1.3 多重 t 检验

在上例中，单因素方差分析是从总体的角度上说明各效应的均值之间存在显著差异，但具体哪些水平下的均值存在较大差异无从得知，所以我们要对每一对样本均值进行一一比较，即要进行均值的多重比较。

最常用也是最直接的方法是多重 t 检验方法，其对因子 A 每个水平下的数据两两比较均值进行 t 检验，但估计方差时用的是全部数据的误差均方和 MS_E 。检验的假设为

$$H_0: \mu_i = \mu_j, \quad i \neq j, \quad i, j = 1, 2, \dots, r$$

构造 t 检验统计量

$$T_{ij} = \frac{\bar{X}_i - \bar{X}_j}{\sqrt{MS_E \left(\frac{1}{n_i} + \frac{1}{n_j} \right)}}, \quad i \neq j, \quad i, j = 1, 2, \dots, r$$

在原假设成立的情况下，该统计量服从自由度为 $n-r$ 的 t 分布，所以检验拒绝域为 $|T_{ij}| > t_{\alpha/2}(n-r)$ 。

多重 t 检验的思路简单，但这样的方法并不能直接使用，这是因为在样本之间多次重复使用 t 检验会大大增加犯第一类错误的概率。例如因子 A 有三个水平，需要比较 3 次 ($n=k(k-1)/2$)，显著性水平为 0.05，所以每次比较犯第一类错误的概率即 0.05，那么 3 次比较同时进行，犯第一类错误的总概率为 $1-(1-\alpha)^n = 1-0.95^3 = 0.1426$ ，所以简单的多重 t 检验结果是不一定可靠的。

这样由多重 t 检验计算的 P 值不能直接用于作判断, 统计学家们提供了很多方法对 P 值进行修正, R 中的 P 值修正函数是 `p.adjust()`。其调用格式为

```
p.adjust(p, method = p.adjust.methods, n = length(p))
```

在 R 中输入如下指令可以看到调整方法的列表 (如表 8.3 所示)。

```
> p.adjust.methods
[1] "holm"      "hochberg"  "hommel"    "bonferroni" "BH"        "BY"        "fdr"
[8] "none"
```

表 8.3 P 值调整方法

调整方法	对应的 R 软件参数
Bonferroni	"bonferroni"
Holm(1979)	"holm"
Hochberg(1988)	"hochberg"
Hommel(1988)	"hommel"
Benjamini & Hochberg(1995)	"BH"
Benjamini & Yekutieli(2001)	"BY"

当多重检验次数较多时, `bonferroni` 修正方法的效果比较好, 思路也很简单: 如果在同一个数据集上同时进行 n 个独立的假设检验, 那么用于每一假设的统计显著水平, 应为仅检验一个假设时显著水平的 $1/n$, 即 $\alpha' = \alpha/n$, n 为多重 t 检验的次数。

在 R 中计算均值的多重 t 检验, 用函数 `pairwise.t.test()` 返回多重比较的修正 P 值, 其调用格式为

```
pairwise.t.test(x, g, p.adjust.method=p.adjust.methods, pool.sd=!paired, paired=FALSE,
  alternative = c("two.sided", "less", "greater"),...)
```

其中, 参数 x 是响应向量; g 是因子向量; `p.adjust.methods` 给出 P 值的修正方法, 默认按 Holm 方法修正, 若不作任何调整则设为 `none`; `paired` 为逻辑值, 指示是否要配对 t 检验; `alternative` 指明检验的方向问题。

对银行账户的例子作多重 t 检验, 使用 `bonferroni` 修正方法, 可以返回各因子水平下两两检验的 P 值矩阵。

```
> attach(account)
> pairwise.t.test(x,A,p.adjust.method="bonferroni")

Pairwise comparisons using t tests with pooled SD
data:  x and A
      1      2
2 0.0013  -
3 3.9e-07 6.5e-10

P value adjustment method: bonferroni
```

经过修正后的 P 值比原来会增大很多，这在一定程度上克服了多重 t 检验增加犯第一类错误的概率的缺点。从检验结果来看，样本两两之间 t 检验的 P 值都很小，说明几个样本之间差异明显。

8.1.4 Kruskal-Wallis 秩和检验

方差分析是关注三个或更多总体的均值是否相等的问题，数据被假设成具有正态分布和齐方差性，这样 F 检验才能奏效。但有时采集的数据并不能完全满足这些条件，类似两两样本的比较，我们不妨尝试将数据转换成秩统计量，因为秩统计量的分布与总体分布无关，可以摆脱总体分布的束缚。在比较两个以上的总体时，可广泛使用非参数的 Kruskal-Wallis 秩和检验，它是对两个以上的秩样本进行比较。

Kruskal-Wallis 秩和检验，首先要求从总体中抽取的样本是独立的，然后将所有样本的值混合在一起看成是单一样本，再把这个单一的混合样本中值从小到大排序，序列值替换成“秩”，最小的值给予秩值 1，有结点时平分秩值。将数据样本转换成秩样本后，再对这个秩样本进行方差分析，但需要注意一点：此时我们构造的统计量 KW 不等于组间平均平方和除以组内平均平方和，而是组间平方和除以全体样本秩方差，这样可以消除数据量纲的影响。这个 KW 统计量是我们判定各组之间是否存在差异的有力依据。

假设有 k 组样本， n_i 是第 i 组样本中的观察数， n 是所有样本中的观察值个数； R_i 是第 i 组样本中的秩和， R_{ij} 是第 i 组样本中的第 j 个观察值的秩值。为简化计算，假设观测值中无结点，第 i 组的 n_i 个个体的秩满足

$$R_{i1} < R_{i2} < \cdots < R_{in_i} \quad (i = 1, 2, \dots, k)$$

需要检验的原假设为

$$H_0: \text{各处理方法的效果无显著差异}$$

在原假设为真时，各组之间不存在差异，或者说各组样本属于的总体具有相同的中心，即各组样本的秩平均应该与全体样本的秩平均 $\frac{1+2+\cdots+n}{n} = \frac{n+1}{2}$ 比较接近，从而得到刻画这种接近程度的统计量——组间平方和

$$SSR = \sum_{i=1}^k n_i \left(\frac{R_i}{n_i} - \frac{n+1}{2} \right)^2$$

我们最终需要计算出 KW 统计量，它的分子是组间平方和，分母应为全体样本秩方差。样本方差的自由度为 $n-1$ ，因此全体样本的秩方差为

$$\begin{aligned}
\text{全体样本秩方差} &= \frac{1}{n-1} \sum_{i=1}^k \sum_{j=1}^{n_i} \left(R_{ij} - \frac{n+1}{2} \right)^2 \\
&= \frac{1}{n-1} \sum_{i=1}^n \left(i - \frac{n+1}{2} \right)^2 \\
&= \frac{n(n+1)}{12}
\end{aligned}$$

根据上述步骤，最终可以得到 Kruskal-Wallis 秩和统计量 KW

$$KW = \frac{12}{n(n+1)} \sum_{i=1}^k n_i \left(R_{i\cdot} - \frac{n+1}{2} \right)^2$$

若样本中存在结点，需要对上式进行调整，首先计算一个矫正系数 C

$$C = 1 - \frac{\sum_j (\tau_j^3 - \tau_j)}{n^3 - n}$$

其中 τ_j 为第 j 个结点的个数，调整后的 KW_c 统计量为

$$KW_c = KW / C$$

如果每组样本中的观察数目至少有 5 个，那么样本统计量 KW_c 非常接近自由度为 $k-1$ 的卡方分布。因此，我们将用卡方分布来决定 KW_c 统计量的检验。 KW 值越大，拒绝原假设的可能性越高。在 R 中主要依靠计算的 P 值来判断：当 P 值小于相应的显著性水平时，拒绝原假设。

R 软件的内置函数 `kruskal.test()` 可以完成 Kruskal-Wallis 秩和检验，使用方法如下：

```
kruskal.test(x, ...)
kruskal.test(x, g, ...)
kruskal.test(formula, data, subset, na.action, ...)
```

其中， x 是数据的向量或列表； g 是由因子构成的向量，但当 x 是列表形式时忽略参数 g 。另外，也可以使用公式形式来设置参数，`formula` 即方差分析的公式；`data` 指定样本数据框；如果只需要 `data` 的一部分子集参与计算，则用 `subset` 指定；`na.action` 设置遇到缺失值时应采取的行为。

某制造商雇用了来自三所本地大学的雇员作为管理人员。最近，公司的人事部门已经收集信息并考核了年度工作成绩。从三所大学来的雇员中随机地抽取了三个独立样本，样本量分别为 7、6、7，数据如表 8.4 所示。制造商想知道来自这三所不同的大学的雇员在管理岗位上的表现是否有所不同，我们通过 Kruskal-Wallis 秩和检验来得到结论。

表 8.4 三所大学的雇员的工作成绩

雇员编号	大学 A	大学 B	大学 C
1	25	60	50
2	70	20	70
3	60	30	60
4	85	15	80
5	95	40	90
6	90	35	70
7	80		75

检验的原假设为

$$H_0: \text{来自三所大学的雇员表现无显著差异}$$

首先在 R 中输入数据，再调用函数 `kruskal.test()` 作检验。数据输入时用到了函数 `factor()`，将数值型变量转换为因子型变量。

```
> data=data.frame(x=c(25,70,60,85,95,90,80,60,20,30,15,40,35,50,70,60,80,90,70,75),
g=factor(rep(1:3,c(7,6,7))))
> kruskal.test(x~g, data=data)
>#另一种写法: kruskal.test(data$x,data$g)
```

```
Kruskal-Wallis rank sum test
```

```
data: x by g
```

```
Kruskal-Wallis chi-squared = 8.9839, df = 2, p-value = 0.0112
```

检验的结果为 $P=0.0112<0.05$ ，因此拒绝原假设，说明来自这三个不同的大学的雇员在管理岗位上的表现有比较显著的差异。

8.2 双因素方差分析及 R 实现

在实际应用中，一个试验的指标往往受到多个因素的影响，除了这些因素本身，因素不同水平的搭配也会影响试验结果。

例如饮料的销售量有可能受销售地区的影响，人们在购买时还会关心饮料颜色。在方差分析中，若把饮料的颜色看作影响销售量的因素 A ，把饮料的销售地区看作影响因素 B 。同时对因素 A 和因素 B 进行分析，就称为双因素方差分析。两个因素的不同水平交叉搭配也有可能对指标产生影响，根据这一点双因素的方差分析主要可分为两类，我们先来讨论无交互作用的情况。

8.2.1 无交互作用的分析

设因素 A 有 r 个水平, B 有 s 个水平, 在每一个水平组合 (A_i, B_j) 下进行一次独立试验得到观测值为 $x_{ij}, i=1, 2, \dots, r; j=1, 2, \dots, s$, 从而双因素方差分析的数据结构如表 8.3 所示。

表 8.5 双因素方差分析的数据结构

	B ₁	B ₂	...	B _s
A ₁	x ₁₁	x ₁₂	...	x _{1s}
A ₂	x ₂₁	x ₂₂	...	x _{2s}
⋮	⋮	⋮	⋮	⋮
A _r	x _{r1}	x _{r2}	...	x _{rs}

假定 $x_{ij} \sim N(\mu_{ij}, \sigma^2)$ 且相互独立, 则无交互作用的双因素方差分析可以表示为如下的数学模型

$$\begin{cases} x_{ij} = \mu + \alpha_i + \beta_j + \varepsilon_{ij}; & i=1, 2, \dots, r, j=1, 2, \dots, s \\ \varepsilon_{ij} \sim N(0, \sigma^2) & \text{且相互独立} \\ \sum_{i=1}^r \alpha_i = 0, & \sum_{j=1}^s \beta_j = 0 \end{cases}$$

其中, $\mu = \frac{1}{rs} \sum_{i=1}^r \sum_{j=1}^s \mu_{ij}$ 为全部数据的总平均, α_i 是因素 A 的第 i 个水平的效应, β_j 为因素 B 的第 j 个水平的效应。

与单因素方差分析类似, 在给定的显著性水平 α 下, 分别对因素 A 和 B 提出如下假设

$$H_{01}: \alpha_1 = \alpha_2 = \dots = \alpha_r = 0 (\text{因素 } A \text{ 对指标无显著影响})$$

$$H_{02}: \beta_1 = \beta_2 = \dots = \beta_s = 0 (\text{因素 } B \text{ 对指标无显著影响})$$

双因素方差分析与单因素方差分析的基本原理相同, 也是基于对总方差的分解

$$SS_T = SS_A + SS_B + SS_E$$

其中总离差平方和为 $SS_T = \sum_{i=1}^r \sum_{j=1}^s (x_{ij} - \bar{x})^2$, $\bar{x} = \frac{1}{rs} \sum_{i=1}^r \sum_{j=1}^s x_{ij}$ 。 SS_A 为因素 A 的效应平方和, SS_B 为因素 B 的效应平方和, SS_E 为误差平方和。

$$SS_A = s \sum_{i=1}^r (\bar{x}_{i\cdot} - \bar{x})^2, \quad \bar{x}_{i\cdot} = \frac{1}{s} \sum_{j=1}^s x_{ij}, \quad i = 1, 2, \dots, r$$

$$SS_B = r \sum_{j=1}^s (\bar{x}_{\cdot j} - \bar{x})^2, \quad \bar{x}_{\cdot j} = \frac{1}{r} \sum_{i=1}^r x_{ij}, \quad j = 1, 2, \dots, s$$

$$SS_E = \sum_{i=1}^r \sum_{j=1}^s (x_{ij} - \bar{x}_{i\cdot} - \bar{x}_{\cdot j} + \bar{x})^2$$

① 在原假设 H_{01} 的条件下，经过统计分析可以得出

$$\frac{SS_A}{\sigma^2} \sim \chi^2(r-1), \quad \frac{SS_E}{\sigma^2} \sim \chi^2((r-1)(s-1))$$

并且 SS_A 与 SS_E 相互独立，于是可以构造出 F 统计量

$$F_A = \frac{SS_A/(r-1)}{SS_E/[(r-1)(s-1)]} \sim F(r-1, (r-1)(s-1))$$

② 同样地，在原假设 H_{02} 的条件下，对因素 B 也可以得出同样的 F 统计量

$$\frac{SS_B}{\sigma^2} \sim \chi^2(s-1), \quad F_B = \frac{SS_B/(s-1)}{SS_E/[(r-1)(s-1)]} \sim F(s-1, (r-1)(s-1))$$

把上面的分析过程汇总为双因素方差分析表，如表 8.6 所示。

表 8.6 双因素方差分析表

方差来源	自由度	平方和	均方	F 值	P 值
因素 A	$r-1$	SS_A	$MS_A = \frac{SS_A}{r-1}$	$F_A = \frac{MS_A}{MS_E}$	P_A
因素 B	$s-1$	SS_B	$MS_B = \frac{SS_B}{s-1}$	$F_B = \frac{MS_B}{MS_E}$	P_B
误差	$(r-1)(s-1)$	SS_E	$MS_E = \frac{SS_E}{(r-1)(s-1)}$		
总和	$rs-1$	SS_T			

R 仍然用函数 `aov()` 作双因素方差分析，只需将 `formula` 改为 $x \sim A + B$ 的形式即可。

例如，某种饮料有五种不同的包装方式，分别在五个不同地区销售。现从每个地区随机抽取一个规模相同的超市，得到该饮料不同包装的销售资料如表 8.7 所示。

表 8.7 某商品在不同地区、不同包装的销售数据

		包装方式(A)				
		A_1	A_2	A_3	A_4	A_5
销	B_1	20	12	20	10	14
售	B_2	22	10	20	12	6
地	B_3	24	14	18	18	10
区	B_4	16	4	8	6	18
(B)	B_5	26	22	16	20	10

首先为了建立数据集，引入生成因子水平的函数 `gl()`，其调用格式为

```
gl(n, k, length = n*k, labels = 1:n, ordered = FALSE)
```

n 是因子的水平个数； k 表示每一水平上的重复次数；`length=n*k` 表示总观测数；可通过参数 `labels` 对因子的不同水平添加标签；`ordered` 为逻辑值，指示是否排序。例如在上面的例子中， A 一共有 5 个水平，并且每个水平都要重复出现 5 次，因此 n 和 k 都等于 5。

```
> x=c(20,12,20,10,14,22,10,20,12,6,24,14,18,18,10,16,4,8,6,18,26,22,16,20,10)
> sales=data.frame(x,A=gl(5,5),B=gl(5,1,25))
> sales$B
[1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
Levels: 1 2 3 4 5
```

分析前先对因素 A 和 B 作方差齐性检验，使用函数 `bartlett.test()`。

```
> bartlett.test(x~A,data=sales)

Bartlett test of homogeneity of variances
data: x by A
Bartlett's K-squared = 0.6653, df = 4, p-value = 0.9555
> bartlett.test(x~B,data=sales)

Bartlett test of homogeneity of variances
data: x by B
Bartlett's K-squared = 1.2046, df = 4, p-value = 0.8773
```

因素 A 和 B 的 P 值都远大于 0.05 的显著性水平，不能拒绝原假设，说明因素 A 、 B 的各水平是满足方差齐性的。这时再进行双因素方差分析，输入指令

```
> sales.aov=aov(x~A+B,data=sales)
> summary(sales.aov)

      Df Sum Sq Mean Sq F value    Pr(>F)    
A         4   199.4    49.84    2.303  0.1032    
B         4   335.4    83.84    3.874  0.0219 *  
Residuals  16   346.2    21.64                 
...
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

检验的结论：因素 B 的 P 值=0.0219<0.05，拒绝原假设，说明销售地区对饮料的销售量有显著影响；而因素 A 的 P 值=0.1032>0.05，不能拒绝原假设，因此没有充分的理由可以说明包装方式对销售有明显影响。

8.2.2 有交互作用的分析

有交互作用的双因素方差分析是假定因素 A 和因素 B 的结合会产生出一种新的效应。例如，某一地区的消费者对某种颜色有着与其他地区消费者不同的特殊偏爱，这就是两个因素结合后产生的新效应，属于有交互作用的背景。交互作用的效应体现在因素 A 和 B 之间的搭配对试验结果产生影响，所以两因素 A 、 B 的不同水平的搭配必须作重复试验。在每个水平组合 (A_i, B_j) 下重复试验 t 次，记第 k 次的观测值为 x_{ijk} ，从而双因素方差分析的数据结构如表 8.8 所示。

表 8.8 双因素方差分析的数据结构

	B_1	B_2	...	B_s
A_1	$x_{111} \ x_{112} \ \dots \ x_{11t}$	$x_{121} \ x_{122} \ \dots \ x_{12t}$...	$x_{1s1} \ x_{1s2} \ \dots \ x_{1st}$
A_2	$x_{211} \ x_{212} \ \dots \ x_{21t}$	$x_{221} \ x_{222} \ \dots \ x_{22t}$...	$x_{2s1} \ x_{2s2} \ \dots \ x_{2st}$
\vdots	\vdots	\vdots	\vdots	\vdots
A_r	$x_{r11} \ x_{r12} \ \dots \ x_{r1t}$	$x_{r21} \ x_{r22} \ \dots \ x_{r2t}$...	$x_{rs1} \ x_{rs2} \ \dots \ x_{rst}$

假定 $x_{ijk} \sim N(\mu_{ij}, \sigma^2)$, $i=1,2,\dots,r$; $j=1,2,\dots,s$; $k=1,2,\dots,t$ 且相互独立，则有交互作用的双因素方差分析可以表示为如下的数学模型

$$\begin{cases} x_{ijk} = \mu + \alpha_i + \beta_j + \delta_{ij} + \varepsilon_{ijk}, & i=1,2,\dots,r; j=1,2,\dots,s; k=1,2,\dots,t \\ \varepsilon_{ijk} \sim N(0, \sigma^2) \text{ 且相互独立} \\ \sum_{i=1}^r \alpha_i = 0, \sum_{j=1}^s \beta_j = 0, \sum_{i=1}^r \delta_{ij} = \sum_{j=1}^s \delta_{ij} = 0 \end{cases}$$

其中， $\mu = \frac{1}{rs} \sum_{i=1}^r \sum_{j=1}^s \mu_{ij}$ 为全部数据的总平均， α_i 是因素 A 的第 i 个水平的效应， β_j 为因素 B 的第 j 个水平的效应， δ_{ij} 表示 A_i 和 B_j 的交互效应。

在给定的显著性水平 α 下，检验的假设为

$$H_{01} : \alpha_1 = \alpha_2 = \dots = \alpha_r = 0 \text{ (因素 } A \text{ 对指标无显著影响)}$$

$$H_{02} : \beta_1 = \beta_2 = \dots = \beta_s = 0 \text{ (因素 } B \text{ 对指标无显著影响)}$$

$$H_{03} : \delta_{11} = \delta_{12} = \dots = \delta_{1s} = 0 \text{ (因素 } A \text{ 和 } B \text{ 无交互作用)}$$

方差分析与前面的原理相同，对总方差作如下分解

$$SS_T = SS_A + SS_B + SS_{A \times B} + SS_E$$

其中总离差平方和为

$$SS_T = \sum_{i=1}^r \sum_{j=1}^s \sum_{k=1}^t (x_{ijk} - \bar{x})^2, \quad \bar{x} = \frac{1}{rst} \sum_{i=1}^r \sum_{j=1}^s \sum_{k=1}^t x_{ijk}$$

SS_A 为因素 A 的效应平方和, SS_B 为因素 B 的效应平方和, $SS_{A \times B}$ 为 A 和 B 的交互效应平方和, SS_E 为误差平方和。

$$SS_A = st \sum_{i=1}^r (\bar{x}_{i..} - \bar{x})^2, \quad \bar{x}_{i..} = \frac{1}{st} \sum_{j=1}^s \sum_{k=1}^t x_{ijk}, \quad i = 1, 2, \dots, r$$

$$SS_B = rt \sum_{j=1}^s (\bar{x}_{.j.} - \bar{x})^2, \quad \bar{x}_{.j.} = \frac{1}{rt} \sum_{i=1}^r \sum_{k=1}^t x_{ijk}, \quad j = 1, 2, \dots, s$$

$$SS_{A \times B} = t \sum_{i=1}^r \sum_{j=1}^s (\bar{x}_{ij.} - \bar{x}_{i..} - \bar{x}_{.j.} + \bar{x})^2$$

$$SS_E = \sum_{i=1}^r \sum_{j=1}^s \sum_{k=1}^t (x_{ijk} - \bar{x}_{ij.})^2, \quad \bar{x}_{ij.} = \frac{1}{t} \sum_{k=1}^t x_{ijk}$$

① 在原假设 H_{01} 的条件下, 经过统计分析可以构造出 F 统计量

$$F_A = \frac{SS_A / (r-1)}{SS_E / [rs(t-1)]} \sim F(r-1, rs(t-1))$$

② 同样地, 在原假设 H_{02} 的条件下, 对因素 B 也可以得出同样的 F 统计量

$$F_B = \frac{SS_B / (s-1)}{SS_E / [rs(t-1)]} \sim F(s-1, rs(t-1))$$

③ 当 H_{03} 成立时,

$$F_{A \times B} = \frac{SS_{A \times B} / [(r-1)(s-1)]}{SS_E / [rs(t-1)]} \sim F((r-1)(s-1), rs(t-1))$$

把上面的分析过程汇总为双因素方差分析表, 如表 8.9 所示。

表 8.9 有交互作用的双因素方差分析表

方差来源	自由度	平方和	均方	F 值	P 值
因素 A	$r-1$	SS_A	$MS_A = \frac{SS_A}{r-1}$	$F_A = \frac{MS_A}{MS_E}$	P_A
因素 B	$s-1$	SS_B	$MS_B = \frac{SS_B}{s-1}$	$F_B = \frac{MS_B}{MS_E}$	P_B
交互作用 $A \times B$	$(r-1)(s-1)$	$SS_{A \times B}$	$MS_{A \times B} = \frac{SS_{A \times B}}{(r-1)(s-1)}$	$F_{A \times B} = \frac{MS_{A \times B}}{MS_E}$	$P_{A \times B}$
误差	$rs(t-1)$	SS_E	$MS_E = \frac{SS_E}{(r-1)(s-1)}$		
总和	$rs-1$	SS_T			

R 仍然用函数 `aov()` 作双因素方差分析, 只需将 formula 改为 $x \sim A + B + A:B$ 或 $x \sim A*B$ 的形式即可。

城市道路交通管理部门为了研究不同路段、不同时段의 拥堵情况, 分别在三个路段和高峰期、非高峰期进行试验, 每一个水平组合下测量 5 次, 共获得 30 个行车时间的数据, 如表 8.10 所示。

表 8.10 不同路段和不同时段의 行车时间数据

	路段 I	路段 II	路段 III
高峰期	25 24 27 25 25	19 20 23 22 21	29 28 31 28 30
非高峰期	20 17 22 21 17	18 17 13 16 12	22 18 24 21 22

首先构造数据集, 对因素 A 和 B 作方差齐性检验, 利用函数 `bartlett.test()`。

```
> time=c(25,24,27,25,25,19,20,23,22,21,29,28,31,28,30,20,17,22,21,17,18,17,13,16,1
2,22,18,24,21,22)
> traffic=data.frame(time,A=gl(2,15,30),B=gl(3,5,30,labels=c("I","II","III")))
> bartlett.test(time~A,data=traffic)
```

```
Bartlett test of homogeneity of variances
data: time by A
Bartlett's K-squared = 0.0533, df = 1, p-value = 0.8174
> bartlett.test(time~B,data=traffic)
```

```
Bartlett test of homogeneity of variances
data: time by B
Bartlett's K-squared = 0.5776, df = 2, p-value = 0.7492
```

检验结果的 P 值均远大于显著性水平 0.05, 说明两个因素下的各水平都满足方差齐性的要求, 可以进一步做方差分析。我们先画图来观察一下数据的特点, 首先是箱线图。

```
> op=par(mfrow=c(1,2)) #分割图形区域
> plot(time~A+B,data=traffic)
```


Hit <Return> to see next plot:

绘制结果如图 8.2 所示。

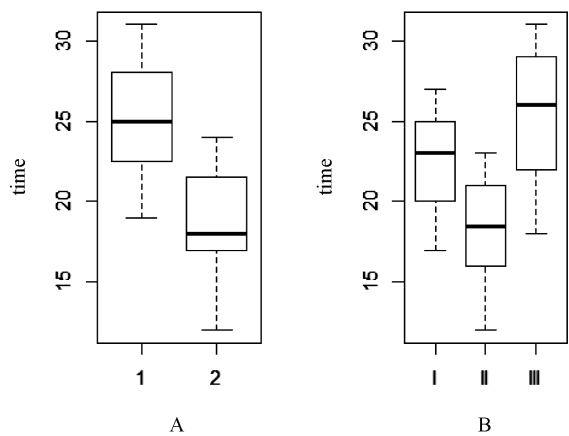


图 8.2 时段和路段的单独效应

从图形上单独观察时段和路段对行车时间的影响，可以发现因素的不同水平还是有明显差别的。为了考察因素间的交互作用是否存在，利用函数 `interaction.plot()` 绘制交互效应图，这是一个高级绘图函数。

```
interaction.plot(x.factor, trace.factor, response, fun = mean, type = c("l", "p", "b", "o", "c"),
                legend = TRUE, trace.label = deparse(substitute(trace.factor)),...)
```

其中，参数 `x.factor` 表示横轴的因子；`trace.factor` 表示分类绘图的因子；`response` 是数值向量，要输入响应变量；`fun` 表示汇总数据的方式，默认为计算每个因子水平下的均值；`type` 指定图形类型；`legend` 是逻辑值，指示是否生成图例；`trace.label` 给出图例中的标签。

```
> attach(traffic)
> interaction.plot(A,B,time,legend=F)
> interaction.plot(B,A,time,legend=F)
```

绘制结果如图 8.3 所示。

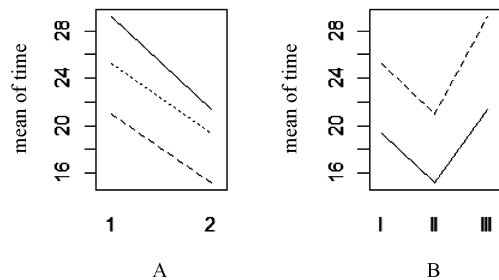


图 8.3 时段和路段的单独效应

图 8.3 中的曲线均没有相交, 所以可以初步判断两个因素之间应该没有交互作用。在图形判断的基础上, 我们用方差分析进行确认。

```
> traf.aov=aov(time~A*B,data=traffic)
> summary(traf.aov)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
A	1	313.63	313.63	84.766	2.41e-09 ***
B	2	261.60	130.80	35.351	7.02e-08 ***
A:B	2	6.67	3.33	0.901	0.42
Residuals	24	88.80	3.70		

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

根据检验结果的 P 值作判断: 因素 A 时段和 B 路段对行车时间有显著影响; 而交互作用 $A:B$ 的 P 值 $=0.42 > 0.05$, 因此不能拒绝原假设 H_{03} , 说明两个因素间没有明显的交互效应。

8.3 协方差分析及 R 实现

为了提高试验的精确性和准确性, 我们对除研究因素以外的一切条件都需要采取有效措施严加控制, 使它们在因素的不同水平间尽量保持一致, 这叫做试验控制。但当我们进行试验设计时, 即使做出很大努力控制, 也经常会碰到试验个体的初始条件不同的情况, 如果不考虑这些因素有可能导致结果失真。如果考虑这些不可控的因素, 这种方差分析就叫做协方差分析, 其是将回归分析和方差分析结合在一起的方法。它的基本原理如下: 将一些对响应变量 Y 有影响的变量 X (未知或难以控制的因素) 看作协变量, 建立响应变量 Y 随 X 变化的线性回归分析, 从 Y 的总的平方和中扣除 X 对 Y 的回归平方和, 对残差平方和作进一步分解后再进行方差分析。

我们从简单的情形入手, 讨论只有一个协变量和一个因子的协方差分析。设试验中的因子为 A , 有 r 个水平 A_1, A_2, \dots, A_r , 在每个水平下进行试验得到 n_i 对观测数据 $(x_{ij}, y_{ij}), i=1, 2, \dots, r; j=1, 2, \dots, n_i$, 则协方差的数学模型为

$$\begin{cases} y_{ij} = \mu + \alpha_i + \beta(x_{ij} - \bar{x}) + \varepsilon_{ij}, i=1, 2, \dots, r, j=1, 2, \dots, n_i \\ \varepsilon_{ij} \sim N(0, \sigma^2) \text{ 且相互独立} \\ \sum_{i=1}^r n_i \alpha_i = 0, \beta \neq 0 \end{cases}$$

其中 μ 是所有数据的总均值, α_i 为第 i 个水平的效应, β 是 y 对 x 的线性回归系数, ε_{ij} 是随机误差。

在给定的显著性水平 α 下, 考虑检验的原假设

$$H_0: \alpha_1 = \alpha_2 = \dots = \alpha_r = 0, \quad H_1: \alpha_1, \alpha_2, \dots, \alpha_r \text{ 不全相等}$$

根据一元线性回归的参数估计（详见下章），可以得出参数的估计值

$$\hat{\mu} = \bar{y}, \quad \hat{\beta} = \frac{SP_E}{SS_E(x)}, \quad \hat{\alpha}_i = \bar{y}_{i\cdot} - \bar{y} - \hat{\beta}(\bar{x}_{i\cdot} - \bar{x})$$

其中， $SS_E(x)$ 为 x 的组内平方和， SP_E 为 x 和 y 的组内乘积和，表示为

$$SP_E = \sum_{i=1}^r \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_{i\cdot})(y_{ij} - \bar{y}_{i\cdot})$$

协方差分析是结合了线性回归的方差分析， $\hat{\beta}(\bar{x}_{i\cdot} - \bar{x})$ 反映了线性回归对数据的矫正，矫正后的组内平方和为

$$SS_E = SS_E(y) - \hat{\beta} \cdot SP_E = SS_E(y) - \frac{SP_E^2}{SS_E(x)}$$

它是 σ^2 的一个无偏估计量，从而可以构造服从自由度 $n-r-1$ 的卡方分布统计量，即

$$\frac{SS_E}{\sigma^2} \sim \chi^2(n-r-1)$$

类似地，矫正后的总平方和为 $SS_T = SS_T(y) - \frac{SP_T^2}{SS_T(x)}$ ，从而得到组间平方和

$$SS_A = SS_T - SS_E, \quad \text{通过它构造服从自由度 } r-1 \text{ 的卡方分布统计量 } \frac{SS_A}{\sigma^2} \sim \chi^2(r-1)。$$

从而在原假设下，有 F 统计量

$$F = \frac{SS_A/(r-1)}{SS_E/(n-r-1)} \sim F(r-1, n-r-1)$$

若 $F > F_{\alpha}(r-1, n-r-1)$ ，则拒绝原假设，认为各因素在不同水平下的试验结果有显著差别；反之，则不能拒绝原假设，认为因素对试验结果无显著影响。在统计软件中进行协方差分析可以计算出 P 值，通过 P 值来判断是否拒绝原假设会更加容易。

R 中计算协方差分析的函数是 `ancova()`，它来源于 Heiberger 编写的统计分析程序包 HH，其调用格式为

```
ancova(formula, data.in = NULL, ..., x, groups)
```

通过参数 `formula` 指定模型；`data.in` 指定输入数据框；`x` 为协方差分析中的协变量，在作图时如果 `formula` 中不包括 `x` 则需要指明；`groups` 指明绘图时分组的条件因子，同样是当 `formula` 中“|”

后面不包括变量时使用。

下面看一个例子。为研究三种肥料对苹果树产量的影响有无差异，每种肥料各施用于 8 棵树，测得每棵树的初始产量和增量（如表 8.11 所示），利用协方差分析三种肥料的效果是否相同。

表 8.11 施用 3 种肥料的苹果产量

	肥料	观察值
初始产量	A	15 13 11 12 12 16 14 17
	B	17 16 18 18 21 22 19 18
	C	22 24 20 23 25 27 30 32
产量增量	A	85 83 65 76 80 91 84 90
	B	97 90 100 95 103 106 99 94
	C	89 91 83 95 100 102 105 110

首先将上述数据输入，生成一个数据框。这里用到了函数 `gl()` 生成因子水平，其调用格式为 `gl(n, k, length = n*k, labels = 1:n, ordered = FALSE)`

其中 `n`、`k`、`length` 都是整数，分别指定因子数量、每个水平的重复次数和生成因子向量的长度。

```
> Weight_Initial=c(15,13,11,12,12,16,14,17,17,16,18,18,21,22,19,18,22,24,20,23,25,27,30,32)
> Weight_Increment=c(85,83,65,76,80,91,84,90,97,90,100,95,103,106,99,94,89,91,83,95,100,102,105,110)
> feed=gl(3,8,24) #生成因子向量
> data_feed=data.frame(Weight_Initial,Weight_Increment,feed)
> library(HH)
Loading required package: multcomp
Loading required package: mvtnorm
Loading required package: survival
Loading required package: splines
Loading required package: TH.data
> m=ancova(Weight_Increment~Weight_Initial+feed, data=data_feed)
> m
Analysis of Variance Table

Response: Weight_Increment
          Df Sum Sq Mean Sq F value    Pr(>F)
Weight_Initial  1 1621.12  1621.12 142.445 1.496e-10 ***
feed           2   707.22   353.61  31.071 7.322e-07 ***
Residuals     20   227.61    11.38
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

协方差分析的 P 值非常小，说明结果非常显著，应该拒绝原假设，认为各因素在不同水平下的试验结果有显著差别，即三种肥料对苹果产量有很大的影响。

在 R 中，函数 `ancova()` 不但可以计算出协方差分析结果，还会绘制 ANCOVA 图，给出不同因子水平下的苹果增量，如图 8.4 所示。从图中也可以看出，在 1、2、3 三个不同的因子水平下，苹果产量增量有所不同，B 肥料的苹果增量明显高于 A 肥料，当然，苹果初始产量也是必须考虑的因素之一，所以我们才必须进行协方差分析。根据 1、2、3 的水平位置也可以发现，施加肥料 C 的苹果树初始产量本身就高于施肥料 A 的苹果树。

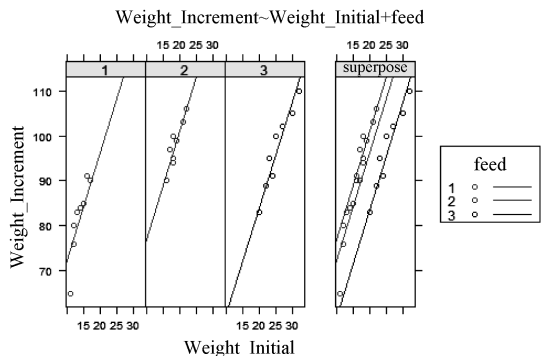


图 8.4 协方差分析图

第 9 章

回归分析及 R 实现

Galton 被誉为现代回归和相关技术的创始人，他在 1875 年进行的豌豆实验中第一次定义了“回归效应”。Galton 挑选了 7 组不同尺寸的豌豆，在英国 7 个不同地区分别种植 10 粒种子，豌豆生长出来后，将原始的豌豆种子（父代）和新生长出来的豌豆（子代）进行比较，来确定豌豆尺寸的遗传规律。最终，他发现结果并非想象中的每一个子代都与父代的尺寸保持一致，恰恰相反，尺寸小的豌豆会得到较大的子代，而尺寸大的豌豆却得到较小的子代。

Galton 当时将这个现象定义为“返祖”，现在这一趋势被称作“回归效应”。若我们把父代和子代看作两个变量，根据上述的规律，找出刻画这两个变量之间因果关系的函数关系，即建立数学模型，就可以根据父代的数值预测子代的取值，这就是经典的回归分析解决的问题。

当然，现代回归分析的应用非常广泛，渗透至各个领域，例如可以描述多个经济因素对 GDP 的影响。本章将通过理论与实例的结合，讲述如何在 R 软件中实现回归分析。

9.1 一元线性回归

9.1.1 模型理论

在回归分析中，被预测或被解释的变量称为因变量（或响应变量） y ，用来预测或解释因变量的一个或多个变量，称为自变量（或解释变量） x 。一元线性回归只涉及一个自变量，回归模型可表示为

$$y = \beta_0 + \beta_1 x + \varepsilon$$

其中， $\beta_0 + \beta_1 x$ 表示 y 随 x 变化而线性变化的部分； ε 是随机误差，包含其他一切不确定影响的总和，通常假定 ε 是一个服从正态分布的随机变量，即 $\varepsilon \sim N(0, \sigma^2)$ 。

当回归模型表示为期望的形式时，被称为回归方程。根据 ε 的假定可知其期望值为 0，因此一元线性回归方程为

$$E(y) = \beta_0 + \beta_1 x$$

其中 β_0 是回归直线在 y 轴上的截距（intercept）； β_1 是直线的斜率。

在做回归分析时，参数 β_0 、 β_1 是未知的，需要利用样本数据去估计它们。估计得到的回归方程表示为

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

估计 $\hat{\beta}_0$ 、 $\hat{\beta}_1$ 的一个直观思想是使得求出的回归直线与各样本观测点之间的偏离越小越好，根据这一思想确定参数的方法即最小二乘法。令

$$Q(\beta_0, \beta_1) = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2$$

那么最小二乘估计就是使 $Q(\beta_0, \beta_1)$ 取到最小值时的 $\hat{\beta}_0$ 、 $\hat{\beta}_1$ 。通过微分方法计算得到

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}, \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

9.1.2 显著性检验

回归分析的主要目的是根据估计的模型用自变量来估计或预测因变量取值，但我们建立的回归方程是否真实地反映了变量之间的相关关系，还需要进一步进行显著性检验。对于一元线性回归模型而言，回归方程的显著性检验有三种等价的方法，分别为 t 检验、 F 检验和相关系数检验。在 R 中给出的方法是 F 检验，原假设为：两个变量之间的线性关系不显著，即 $H_0: \beta_1 = 0$ 。

以回归平方和（SSR）和残差平方和（SSE）为基础，构造 F 统计量为

$$F = \frac{SSR / 1}{SSE / (n - 2)} \sim F(1, n - 2)$$

当给定显著性水平为 α 时，如果检验结果的 P 值小于 α ，则拒绝原假设，说明模型反映的线性关系显著；反之不拒绝原假设。

9.1.3 R 语言实现

在 R 语言中，使用 `lm` 函数可以非常容易地求出回归方程，用它来拟合线性模型，可以进行回归、方差分析和协方差分析。

```
lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE, x = FALSE,
y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ...)
```

lm 函数参数说明见表 9.1。

表 9.1 lm()的参数设置

参数	说明
formula	描述要拟合的回归模型
data	数据框或列表
subset	样本观察值的子集
weights	权重向量，应该是数字向量或 NULL。使用加权最小二乘方法时，weights 是数字向量；若使用普通最小二乘，则为 NULL
na.action	表示数据包含缺失值 NA 时，应该如何处理。默认设置是 na.fail，表示如果数据不包含缺失值就直接返回，否则显示出错。其他设置包括 NULL，表示不对缺失值做处理；na.omit 表示忽略包含缺失值的观察数据；na.pass 表示返回的对象不变。 例如，na.action(na.omit(c(1, NA)))
method	指出拟合的方法。目前仅支持 method = “qr”（QR 分解），method = “model.frame” 返回模型框架（与下面的 method=TRUE 作用相同）
model,x,y,qr	均为逻辑值，如果为 TRUE 返回合适的模型框架、模型矩阵、响应和 QR 分解

除了第一个参数 formula 是必选条目外，其他参数都是可选的。

接下来，我们通过对数据集的回归分析来说明一元线性回归在 R 语言中的实现。表 9.2 中的数据集来源于在埃及的一个村庄卡拉马的真实调查结果，调查者测量了 161 个儿童的身高，每月将这些儿童的平均身高记录下来，形成了这些数据。

表 9.2 一元线性回归原始数据集

平均身高与年龄			
年龄（月）	平均身高（厘米）	年龄（月）	平均身高（厘米）
18	76.1	24	79.9
19	77	25	81.1
20	78.1	26	81.2
21	78.2	27	81.8
22	78.8	28	82.8
23	79.7	29	83.5

首先，为了直观地显示年龄与身高之间的关系，画出一张散点图，以年龄 age 为横坐标，身高 height 为纵坐标，每行的一对数据对应图中的一个点。R 的指令如下：

```
> age=18:29
```



```
> height=c(76.1,77,78.1,78.2,78.8,79.7,79.9,81.1,81.2,81.8,82.8,83.5)
> plot(age,height)
```

绘制结果如图 9.1 所示。

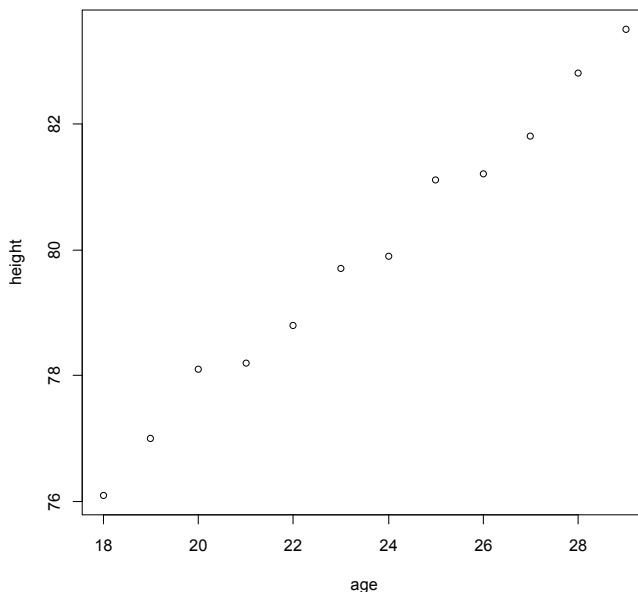


图 9.1 身高与年龄散点图

从图中可以观察到，年龄与身高基本在一条直线附近，可以认为两者具有线性关系，接下来建立回归模型（程序接前文，下同）：

```
> lm.reg=lm(formula=height~age) #可简化为 lm(height~age)
> summary(lm.reg) #提取模型计算结果

Call:
lm(formula = height ~ age)

Residuals:
    Min       1Q   Median       3Q      Max
-0.27238 -0.24248 -0.02762  0.16014  0.47238

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  64.9283     0.5084  127.71 < 2e-16 ***
age           0.6350     0.0214   29.66 4.43e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.256 on 10 degrees of freedom
Multiple R-squared:  0.9888,    Adjusted R-squared:  0.9876
```

```
F-statistic: 880 on 1 and 10 DF, p-value: 4.428e-11
```

指令 `summary` 提供了很多有用的模型结果，包括残差 (Residuals)、回归系数 (Coefficients)、拟合优度 R^2 以及 F 统计量和 P 值。回归系数中 `Intercept` 表示截距 $\hat{\beta}_0=64.9283$ ，`age` 对应的估计值为自变量前的回归系数 $\hat{\beta}_1=0.6350$ ，因此由输出的结果可以得到回归方程 $y = 64.9283 + 0.635x$ 。

我们建立的回归方程仅仅根据最小二乘方法，估计出一个年龄与身高之间的数学关系式，这一式子是否真实地反映了变量之间的相关关系，还需要通过显著性检验结果来说明。从 `summary` 函数输出的结果可以看出，回归系数 $\hat{\beta}_0, \hat{\beta}_1$ 的 P 值非常小，分别为 $P < 2e-16$ 和 $P = 4.43e-11$ ，说明系数是非常显著的（显著程度由后面的 ‘***’ 表示）。而回归方程的 F 检验，得到 P 值为 $4.428e-11$ ，也是非常显著的。上述两种检验，都说明以年龄为自变量构建的回归模型适用于估计身高这一因变量。

为了直观地看出估计效果，我们通过指令 `abline` 在 `plot` 的图像中加入拟合的线性回归线：

```
> abline(lsfrit(age,height)) #可简化为 abline(lm.reg)
```

绘制结果如图 9.2 所示。

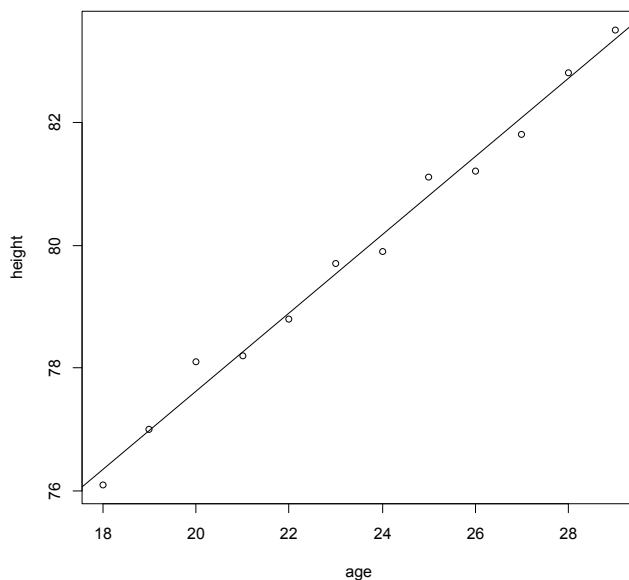


图 9.2 身高与年龄拟合直线

接下来，对残差进行分析，这里介绍两种方法。首先，比较简单的一种是自己画出残差图，在 R 中可用函数 `residuals()` 计算残差：

```
> lm.res=residuals(lm.reg)
> plot(lm.res)
```

得到的残差图如图 9.3 所示。

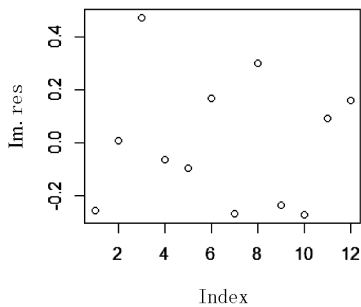


图 9.3 一元线性模型残差图

如果残差图中点比较均匀地分布在 $y=0$ 两侧，则说明残差不包含趋势，只体现随机影响。在图 9.3 中，第 3 个点的残差较大，有些异常，在图 9.2 中也可以看出第 3 个点偏离直线最远。

第二种方法是在回归模型的结果上使用 `plot` 命令，进行误差的诊断检验，由于 `plot(lm.reg)` 命令会输出 4 组图像，首先需要将图像窗口分成 4 部分，在 R 中指令为：

```
> par(mfrow=c(2,2)) #将图像窗口分成 4 份（两行及两列）
> plot(lm.reg)
```

这 4 组图像（如图 9.4 所示）给出了诊断信息，分别表示：

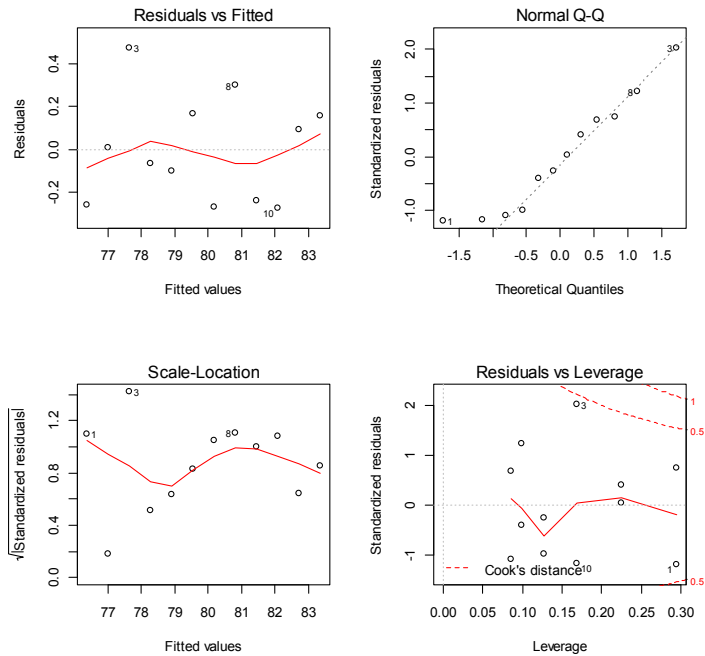


图 9.4 残差诊断图

- 残差图 (Residuals vs Fitted)。横坐标为拟合值 \hat{y} ，纵坐标为残差。从散点图中可以看出，数据点基本均匀地分布在横轴 $y=0$ 两侧时，第 3 个点残差很大。
- 正态分位图 (Normal Q-Q)。Q-Q 图中点的分布集中在 $y=x$ 这条直线上时，说明残差是服从正态分布的。
- 位置-尺度图 (Scale-Location)。纵坐标为标准化残差的平方根，残差越大，点的位置越高。
- 曲式距离图 (或称为残差杠杆图, Residuals vs Leverage)。图中的曲式距离 (Cook's distance plot) 表示每一个数据点对回归线的影响力，第 3 个点的值较大，表示当删除该数据点时，回归系数会有实质上改变，为异常值点。

根据残差分析的结果，我们尝试将第 3 个点从原始数据中剔除，重新拟合回归方程：

```
> age=age[-3];height=height[-3]
> lm.reg2=lm(formula=height~age)
> summary(lm.reg2)

Call:
lm(formula = height ~ age)

Residuals:
    Min       1Q   Median       3Q      Max
-0.27370 -0.17931  0.01957  0.12293  0.32405

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  64.5540     0.4381  147.36 < 2e-16 ***
age           0.6489     0.0182   35.64 5.33e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2073 on 9 degrees of freedom
Multiple R-squared:  0.993,    Adjusted R-squared:  0.9922
F-statistic: 1270 on 1 and 9 DF,  p-value: 5.327e-11
```

剔除异常值点后的回归方程，无论回归系数检验还是 F 检验都更加显著 (P 值变小了)，说明第 3 个样本可以去掉，最终得到的回归方程是 $y = 64.554 + 0.6489x$ 。

最后，根据估计方程，我们可以在给定年龄的情况下，预测对应的身高估计值及其取值区间，取值区间是在给定置信水平 $1-\alpha$ 下的预测区间 $[\hat{y}_0 - \hat{\sigma}z_{1-\alpha/2}, \hat{y}_0 + \hat{\sigma}z_{1-\alpha/2}]$ 。在 R 中，用于预测的指令是 predict 函数：

```
> age.pre<-data.frame(age=30) #predict 函数中使用的数据是 data.frame 或 list 格式
> h.pre<-predict(lm.reg2,age.pre,interval="prediction",level=0.95)
> h.pre
      fit      lwr      upr
```

1 84.02034 83.46839 84.57228

在置信水平 0.95（即默认值）下，当年龄为 30 时，身高的预测值为 84.02034，预测区间为 [83.46839, 84.57228]。

9.2 多元线性回归

我们多次强调，实际问题中经常遇到分析两个或多个变量间关系的情况。当一元回归方程中的自变量增加时，就成为了多元线性回归分析。

9.2.1 模型理论

实际问题中的因变量 Y 通常与多个自变量 $X_1, X_2, \dots, X_p (p > 1)$ 有关，它们之间的线性关系描述为

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p + \varepsilon$$

其中， $\beta_0, \beta_1, \dots, \beta_p$ 是模型的参数； ε 为误差项，反映了随机因素对 Y 的影响，是不能由 X_0, X_1, \dots, X_p 与 Y 之间的线性关系解释的变异性， $\varepsilon \sim N(0, \sigma^2)$ 。设 $(x_{i1}, x_{i2}, \dots, x_{ip}, y_i)$, $i = 1, 2, \dots, n$ 是变量的 n 次独立观测值，从而我们将多元线性回归模型展开

$$\begin{cases} y_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \dots + \beta_p x_{1p} + \varepsilon_1 \\ y_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \dots + \beta_p x_{2p} + \varepsilon_2 \\ \dots\dots\dots \\ y_n = \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \dots + \beta_p x_{np} + \varepsilon_n \end{cases}$$

展开的模型过于烦琐，通常我们将之简化为矩阵形式，令

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}, X = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix}, \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

从而多元线性模型写为 $Y = X\beta + \varepsilon$ 。为计算回归系数的估计值 $\hat{\beta}$ ，首先写出经验回归方程

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \dots + \hat{\beta}_p X_p$$

类似于一元线性回归，使用最小二乘法计算参数 β 的估计值，但在多元情况下这一过程通过矩阵运算完成。观测值与回归方程拟合值之间的残差平方和为

$$Q(\beta) = (y - X\beta)^T (y - X\beta)$$

计算上式的最小值，可以得到求解 $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$ 的标准方程组为

$$\begin{cases} \left. \frac{\partial Q}{\partial \beta_0} \right|_{\beta_0 = \hat{\beta}_0} = 0 \\ \left. \frac{\partial Q}{\partial \beta_i} \right|_{\beta_i = \hat{\beta}_i} = 0 \quad (i = 1, 2, \dots, p) \end{cases}$$

从而得到当 X 为列满秩时， β 的最小二乘估计为 $\hat{\beta} = (X^T X)^{-1} X^T y$ ，残差向量为 $\hat{\varepsilon} = y - X\hat{\beta}$ ， σ^2 的最小二乘估计为 $\hat{\sigma}^2 = \frac{\hat{\varepsilon}^T \hat{\varepsilon}}{n - p - 1}$ 。

9.2.2 显著性检验

一元线性回归方程的 F 检验与回归系数的 t 检验是等价的，因为方程中只有一个自变量，如果 F 检验表明因变量和自变量有着显著的线性关系，那么回归系数必然不为 0。但在多元线性回归中，这两种检验并不一致。在 p 个自变量中，只要有一个自变量与因变量的线性关系显著， F 检验就会通过，但这并不能说明每个自变量在线性模型中影响都显著，所以回归系数 t 检验是必要的，用于判断每个自变量对因变量的影响是否都显著。

（1）回归方程线性关系的检验

回归方程的 F 检验是总体的显著性检验，提出假设为

$$H_0: \beta_0 = \beta_1 = \dots = \beta_p = 0, \quad H_1: \beta_0, \beta_1, \dots, \beta_p \text{ 至少有一个不为 } 0$$

计算 F 检验的统计量

$$F = \frac{SS_R/p}{SS_E/(n-p-1)} \sim F(p, n-p-1)$$

其中， $SS_R = \sum (\hat{y}_i - \bar{y})^2$ 为回归平方和， $SS_E = \sum (y_i - \hat{y}_i)^2$ 为残差平方和。最后作出统计决策，在给定的显著性水平 $\alpha = 0.05$ 下， F 检验的拒绝域为 $F > F_\alpha(p, n-p-1)$ 。同样，在 R 中我们直接利用 P 值作出决策。

（2）回归系数的检验

回归方程通过整体显著性检验后，我们对各回归系数单独进行检验，首先提出假设

$$H_0: \beta_i = 0, \quad H_1: \beta_i \neq 0 \quad (i = 1, 2, \dots, p)$$

计算检验的统计量

$$t_i = \frac{\hat{\beta}_i}{s_{\hat{\beta}_i}} \sim t(n-p-1)$$

其中, $s_{\hat{\beta}_i}$ 是回归系数 $\hat{\beta}_i$ 的标准差估计值, 即

$$s_{\hat{\beta}_i} = \frac{s_y}{\sqrt{\sum x_i^2 - \frac{1}{n}(\sum x_i)^2}} = \frac{\sqrt{SS_E/n-p-1}}{\sqrt{\sum x_i^2 - \frac{1}{n}(\sum x_i)^2}}$$

根据 t 检验作出统计决策: 在给定的显著性水平 α 下, 若 $|t| > t_{\alpha/2}$, 则拒绝原假设; 若 $|t| < t_{\alpha/2}$, 则没有充分的理由拒绝原假设。

9.2.3 R 语言实现

多元线性回归分析同样由函数 `lm()` 完成, 但参数 `formula` 的表达式应表示为多元形式, R 语言中多元关系的表达式总结见表 9.3。

表 9.3 R 中多元关系的表达形式

方程式	含义
<code>y~x</code>	<code>x</code> 为自变量, 一元线性关系
<code>y~x+z</code>	<code>x</code> 、 <code>z</code> 为自变量, 探讨 <code>x</code> 、 <code>z</code> 的主效应
<code>y~x+z+x:z</code>	探讨 <code>x</code> 、 <code>z</code> 的主效应及其交互作用
<code>y~x*z</code>	同 <code>x+z+x:z</code>
<code>y~(x+z+m)^2</code>	同 <code>x+z+m+x:z+x:m+z:m</code> , 包括主效应和因子间的交互效应 (<code>^n</code> 表示包含所有 <code>n</code> 阶以下的交互作用)
<code>y~x%in%z</code>	表示 <code>x</code> 包含在 <code>z</code> 中, 即等同于 <code>z+z:x</code>
<code>y~(x+z+m)^2-x:z</code>	表示从 <code>(x+z+m)^2</code> 中去掉 <code>x:z</code>

下面我们用一个计量经济中的实际模型来说明多元线性回归在 R 中的实现。财政收入是一个国家经济系统最重要的部分, 通过分析财政一般收入中的各分项, 能够及时地发现现行税制和政策是否适应经济发展情况、产业结构是否合理等问题, 因此建立一个合理的财政收入模型十分重要。财政收入会受到各种不同因素的影响, 为深入研究, 我们建立计量经济学模型, 以财政收入 y (亿元) 为因变量, 自变量选取如下: 第一产业国内生产总值 x_1 (亿元)、第二产业国内生产总值 x_2 (亿元)、第三产业国内生产总值 x_3 (亿元)、人口数 x_4 (万人)、社会消费品零售总额 x_5 (亿元)、受灾面积 x_6 (万公顷)。数据集如表 9.4 所示。

表 9.4 1990—2009 年财政收入及相关统计数据

年份	y	x_1	x_2	x_3	x_4	x_5	x_6
1990	2937.10	5062.00	7717.40	5888.42	114333	8300.10	38474.00
1991	3149.48	5342.20	9102.20	7337.10	115823	9415.60	55472.00
1992	3483.37	5866.60	11699.50	9357.38	117171	10993.70	51332.00
1993	4348.95	6963.76	16454.43	11915.73	118517	14270.40	48827.00
1994	5218.10	9572.69	22445.40	16179.76	119850	18622.90	55046.00
1995	6242.20	12135.81	28679.46	19978.46	121121	23613.80	45825.00
1996	7407.99	14015.39	33834.96	23326.24	122389	28360.20	46991.00
1997	8651.14	14441.89	37543.00	26988.15	123626	31252.90	53427.00
1998	9875.95	14817.63	39004.19	30580.47	124761	33378.10	50145.00
1999	11444.08	14770.03	41033.58	33873.44	125786	35647.90	49979.50
2000	13395.23	14944.72	45555.88	38713.95	126743	39105.70	54688.00
2001	16386.04	15781.27	49512.29	44361.61	127627	43055.40	52214.60
2002	18903.64	16537.02	53896.77	49898.90	128453	48135.90	46946.10
2003	21715.25	17381.72	62436.31	56004.73	129227	52516.30	54505.80
2004	26396.47	21412.73	73904.31	64561.29	129988	59501.00	37106.26
2005	31649.29	22420.00	87598.09	74919.28	130756	67176.60	38818.23
2006	38760.20	24040.00	103719.54	88554.88	131448	76410.00	41091.41
2007	51321.78	28627.00	125831.36	111351.95	132129	89210.00	48992.35
2008	61330.35	33702.00	149003.44	131339.99	132802	114830.10	39990.03
2009	68518.30	35226.00	157638.78	147642.09	133474	132678.40	47213.69

设多元线性回归模型为

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6$$

在 R 中输入指令

```
> revenue=read.table("d:/data/revenue.txt",header=T) #读取数据
> lm.reg=lm(y~x1+x2+x3+x4+x5+x6,data=revenue)
> summary(lm.reg) #汇总回归分析结果
```

Call:

```
lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6, data = revenue)
```

Residuals:

```
      Min       1Q   Median       3Q      Max
-295.71 -173.52   26.59   90.16  370.01
```

Coefficients:


```

      Estimate Std. Error  t value Pr(>|t|)
(Intercept)  6.046e+04   3.211e+03   18.829  8.12e-11 ***
x1           -1.171e-01   8.638e-02   -1.356  0.19828
x2            3.427e-02   3.322e-02    1.032  0.32107
x3            6.182e-01   4.103e-02   15.067  1.31e-09 ***
x4           -5.152e-01   2.930e-02  -17.585  1.91e-10 ***
x5           -1.104e-01   2.878e-02   -3.837  0.00206 **
x6           -1.864e-02   1.023e-02   -1.823  0.09143 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```

Residual standard error: 234.8 on 13 degrees of freedom
Multiple R-squared:  0.9999,    Adjusted R-squared:  0.9999
F-statistic: 2.294e+04 on 6 and 13 DF,  p-value: < 2.2e-16

```

计算结果显示, 回归模型的拟合优度 $R^2=0.9999$, 说明模型的拟合效果较好, 但在多元情况下的自变量个数越多, 拟合优度会越高, 还要看检验的结果; 回归方程的 F 检验十分显著 (P 值很小), 回归系数 x_1 、 x_2 不显著, x_6 仅在 0.1 的显著性水平下显著。可以写出回归方程为

$$Y = 60460 - 0.1171X_1 + 0.03427X_2 + 0.6182X_3 - 0.5152X_4 - 0.1104X_5 - 0.01864X_6$$

与一元线性模型相同, 使用函数 `predict()` 可以对以后年份的人口增长率作点预测和区间预测。另外, 还可以通过一些函数获得更多线性拟合模型的信息。表 9.5 中列出了一些这样的函数。

表 9.5 提取线性拟合模型信息的函数

函 数	功 能
<code>coef()</code>	提取系数向量的估计值
<code>resid()/residuals()</code>	提取残差向量
<code>fitted()/predict()</code>	提取拟合值向量
<code>vcov()</code>	提取 β 的普通最小二乘估计量条件方差阵的估计值
<code>deviance()</code>	提取残差平方和
<code>formula()</code>	提取模型公式
<code>df.residual()</code>	提取残差的自由度
<code>nobs()</code>	提取模型中样本个数 n
<code>AIC()</code>	提取模型中 AIC 信息准则
<code>BIC()</code>	提取模型中 BIC 信息准则
<code>logLik()</code>	提取模型的对数似然函数值

其他的一些模型提取函数, 将在下一节回归诊断中详细介绍。

在上面的拟合结果中, 我们发现自变量 x_1 、 x_2 并不显著, 说明第一、二产业国内生产总值对财政收入的解释意义并不显著, 应当从模型中剔除, 最简单的方式是重写拟合模型 `lm.reg=lm(y~x3+x4+x5+x6,data=revenue)`。

R 中的函数 `update()` 是专门用于修正模型的函数, 在原模型的基础上, 不仅可以添加或删除某些项得到新的模型, 还可以对变量进行运算, 如对因变量取对数、开方等。其调用格式为

```
update(object, formula., ..., evaluate = TRUE)
```

`object` 表示已经拟合好的模型对象, 例如存储 `lm()`、`glm()` 的拟合结果; `formula` 指定模型的表达式, 原模型中不变的部分用点 “.” 表示, 只写出需要修正的地方即可, 例如

- `lm.reg1=update(lm.reg,~.+x4)` 表示添加一个新的变量。
- `lm.reg2=update(lm.reg,sqrt(.)~.)` 表示对因变量 Y 作开方运算后再拟合回归模型。

将上例中的 X_3 剔除后重新拟合多元线性回归方程

```
> lm.reg1=update(lm.reg,~.-x1-x2)
> summary(lm.reg1)

Call:
lm(formula = y ~ x3 + x4 + x5 + x6, data = revenue)

Residuals:
    Min       1Q   Median       3Q      Max
-325.62 -147.54   14.07  108.28  427.42

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.339e+04  2.346e+03  27.020 3.89e-14 ***
x3           6.584e-01  1.548e-02  42.523 < 2e-16 ***
x4          -5.438e-01  1.981e-02 -27.445 3.09e-14 ***
x5          -1.392e-01  1.918e-02  -7.256 2.80e-06 ***
x6          -1.803e-02  9.788e-03  -1.842 0.0854 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 233.6 on 15 degrees of freedom
Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
F-statistic: 3.476e+04 on 4 and 15 DF, p-value: < 2.2e-16
```

去除 x_1 、 x_2 后的方程仍然十分显著, 剩余的自变量系数均比较显著, 这时拟合的回归方程为

$$Y = 63390 + 0.6584X_3 - 0.5438X_4 - 0.1392X_5 - 0.01803X_6$$

9.2.4 逐步回归

在实际分析中, 我们使用多元线性模型描述变量之间的关系时, 无法事先了解哪些变量之间的关系显著, 就会考虑很多的潜在自变量。若用上一节的方法一一剔除变量, 建模过程将变得非常烦琐, 所以一般采用逐步回归法。逐步回归建模时, 按偏相关系数的大小次序 (即变量对 y 影响程度) 将自变量逐个引入方程, 对引入的每个自变量的偏相关系数进行统计检验, 效应显著的自变量留在回归方程内, 如此循此继续遴选下一个自变量。

从方法上讲, 逐步回归并没有引入新的理论, 但它是变量选择与建立最优回归的有效方式。何为“最优”? 主要有两个方面: 首先, 自变量个数要适中。若自变量太多, 预测的计算量增大, 并且造成剩余标准差也较大; 其次, 保证自变量的显著性, 对因变量的解释作用越强越好, 所以引入和剔除自变量时都要进行显著性检验, 使之达到最优状态。

R 中进行逐步回归的函数是 `step()`, 以 *AIC* 信息准则作为添加或删除变量的判别方法。*AIC* 准则由日本统计学家赤池弘次创立, 建立在熵的概念基础上, 一般情况 *AIC* 表示为

$$AIC = 2(p+1) - 2\ln(L)$$

其中, p 是回归模型中自变量的个数, L 是似然函数。*AIC* 用于寻找解释性最好且包含最少自由参数的模型, 所以选择变量时优先考虑的模型应是 *AIC* 值最小的那一个。`step()` 的调用格式为

```
step(object, scope, scale = 0, direction = c("both", "backward", "forward"),
      trace = 1, keep = NULL, steps = 1000, k = 2, ...)
```

其中, `object` 是线性模型或广义线性模型的分析结果; `scope` 确定逐步搜索的范围, 是一个公式或包含 `upper`、`lower` 的列表; `direction` 确定逐步回归的方法, 默认值“`both`”指“一切子集回归法”, “`backward`”指后退法, “`forward`”指前进法。`trace` 若是正值, 则逐步回归分析的过程将打印出来; `keep` 是一个过滤器的功能, 通常 `keep` 选择对象元素的一个子集并返回; `steps` 表示回归的最大步数; `k` 指 *AIC* 中的自由度。

对上一节的例子作逐步回归, 每一步的分析都将在结果中显示。

```
> lm.step=step(lm.reg)
Start: AIC=223.73
y ~ x1 + x2 + x3 + x4 + x5 + x6

      Df Sum of Sq      RSS   AIC
- x2    1    58668  775347 223.31
<none>                  716679 223.73
- x1    1   101324  818003 224.38
- x6    1   183147  899826 226.28
- x5    1   811757 1528436 236.88
- x3    1 12515013 13231691 280.05
- x4    1 17048151 17764830 285.94

Step: AIC=223.31
y ~ x1 + x3 + x4 + x5 + x6

      Df Sum of Sq      RSS   AIC
- x1    1    43428  818775 222.40
<none>                  775347 223.31
- x6    1   218417  993764 226.27
- x5    1   1837987 2613334 245.61
- x4    1 22288387 23063734 289.16
- x3    1 97699918 98475265 318.19
```

```

Step: AIC=222.4
y ~ x3 + x4 + x5 + x6

      Df Sum of Sq    RSS   AIC
<none>            818775 222.40
- x6     1    185123  1003898 224.47
- x5     1   2873929  3692704 250.52
- x4     1  41113643 41932417 299.12
- x3     1  98701814 99520589 316.40

```

在默认情况下，逐步回归的每一步过程都会打印出来，本例中逐步回归经历了三步，分别剔除了不显著的自变量 x_1 和 x_2 ， AIC 逐渐减小。最终，R 会选择 AIC 最小的那个模型，即“最优”回归方程。

```

> summary(lm.step)

Call:
lm(formula = y ~ x3 + x4 + x5 + x6, data = revenue)

Residuals:
    Min       1Q   Median       3Q      Max
-325.62 -147.54   14.07  108.28  427.42

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.339e+04   2.346e+03  27.020  3.89e-14 ***
x3           6.584e-01   1.548e-02  42.523  < 2e-16 ***
x4          -5.438e-01   1.981e-02 -27.445  3.09e-14 ***
x5          -1.392e-01   1.918e-02  -7.256  2.80e-06 ***
x6          -1.803e-02   9.788e-03  -1.842  0.0854 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 233.6 on 15 degrees of freedom
Multiple R-squared:  0.9999,    Adjusted R-squared:  0.9999
F-statistic: 3.476e+04 on 4 and 15 DF,  p-value: < 2.2e-16

```

逐步回归直接得到根据 AIC 选择的最优模型，回归方程所有的检验都是显著的，得到的方程为

$$Y = 63390 + 0.6584X_3 - 0.5438X_4 - 0.1392X_5 - 0.01803X_6$$

9.3 回归诊断及 R 实现

回归分析完成后，我们仅从显著性检验的角度了解回归效果，但模型的其他特性还有待商榷，例如异常值、共线性等问题，所以我们应该立即进行回归诊断。本节主要包括三个部分的内容：残差诊断、影响分析以及多重共线性诊断。

9.3.1 残差诊断

通过分析残差我们可以发现数据是否存在异常值。异常值有两种：一种是“真的”，指由于模型的缺陷、数据违背统计假设、特殊个案等因素形成的异常值；还有一种“假的”异常值，是由于失误造成的，比如数据录入错误、计算错误、测量错误等。残差也分为几类：普通残差、标准化残差、学生化残差等。

(1) 普通残差

线性回归模型为 $Y = X\beta + \varepsilon$ ，由回归系数估计值 $\hat{\beta} = (X^T X)^{-1} X^T Y$ 得到 Y 的拟合值为

$$\hat{Y} = X\hat{\beta} = X(X^T X)^{-1} X^T Y = HY$$

其中， $H = X(X^T X)^{-1} X^T$ 称为帽子矩阵，从而残差向量为 $\hat{\varepsilon} = Y - \hat{Y} = (I - H)Y$ 。

利用最小二乘法计算回归模型时，假设中对残差的要求是满足独立性和方差齐性的。所以提取模型残差后，我们要通过画图 and 检验作残差诊断。残差图是以数据序号或 y 的拟合值为横坐标、残差为纵坐标的散点图，常见的图形形状分为如图 9.5 所示的几类。

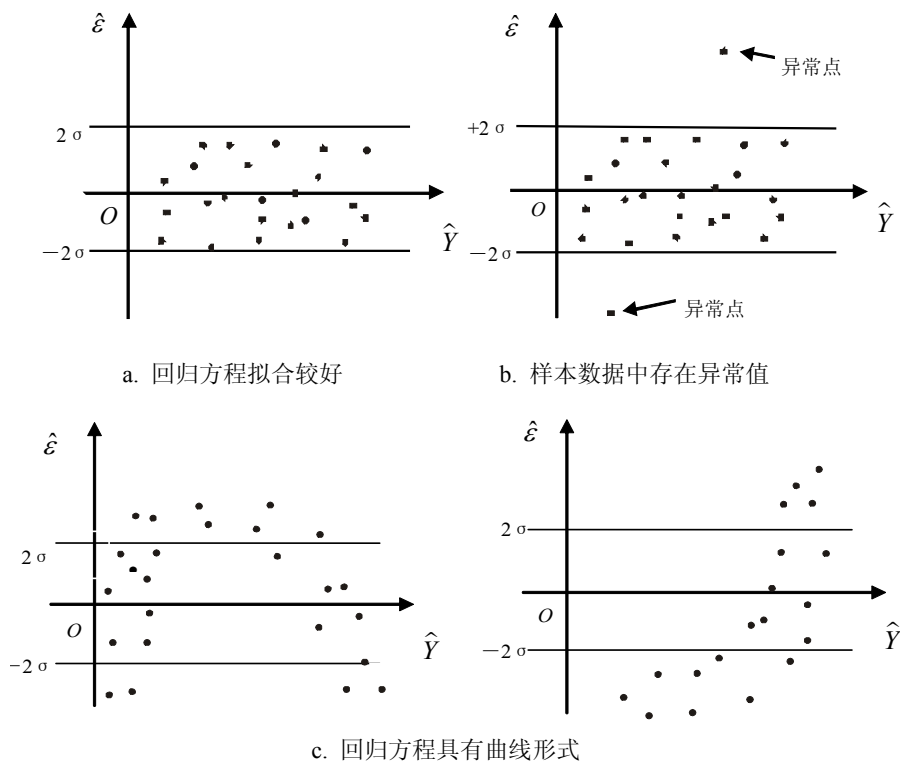
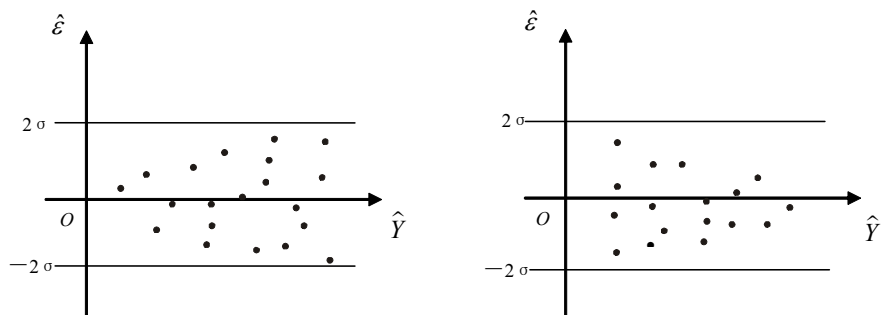
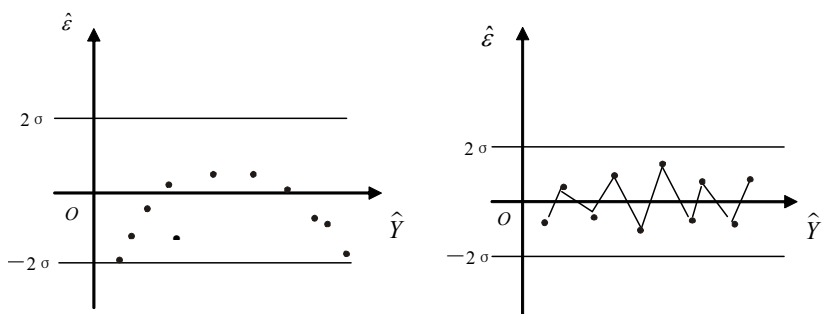


图 9.5 不同类型的残差图



d. 随机误差项存在异方差性



e. 随机误差项存在自相关性

图 9.5 不同类型的残差图（续）

R 中提取残差有两种方式，既可以用前面提到的函数 `residuals()`，也可以用符号 “\$” 直接提取回归分析生成的对象中的元素 `residual`。

我们提取财政收入案例中的模型残差，并绘制残差图，查看残差分布情况。

```
> y.res=lm.reg$residual
> y.fit=predict(lm.reg) #计算 y 的预测值，作为残差图的横坐标
> plot(y.res~y.fit,main="残差图")
```

绘制结果如图 9.6 所示。

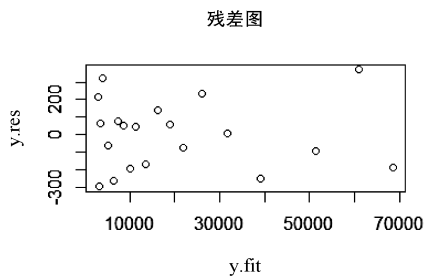


图 9.6 财政收入预测的残差图

从图中可以看出，残差的分布较为均匀，直观上大致满足模型的假设条件，再利用 Shapiro-Wilk 检验判断正态性。

```
> shapiro.test(y.res)

Shapiro-Wilk normality test

data:  y.res
W = 0.9621, p-value = 0.5873
```

上一章介绍过 Shapiro-Wilk 正态检验，从结果可知， P 值=0.5873 远远大于显著性水平 0.05，故不能拒绝原假设，说明数据服从正态分布。

(2) 标准化残差

普通残差与数据的数量级有关，除以标准误差后得到标准化残差。由于

$$E(\hat{\varepsilon}) = 0, \quad Var(\hat{\varepsilon}) = \sigma^2(I - H)$$

因此对每个 $\hat{\varepsilon}$ ，有

$$r_i = \frac{\hat{\varepsilon}_i}{\sigma \sqrt{1 - h_{ii}}} \sim N(0, 1)$$

其中， h_{ii} 是矩阵 H 对角线上的第 i 个元素。 r_i 称为标准化残差，模型拟合较好的情况下将服从标准正态分布。R 中用函数 `rstandard()` 提取标准化残差，调用格式为

```
rstandard(model, infl = lm.influence(model, do.coef = FALSE),
sd = sqrt(deviance(model)/df.residual(model)), ...)
```

参数 `model` 是由 `lm()` 或 `glm()` 生成的对象；`infl` 是由 `lm.influence()` 返回的影响结构；`sd` 是模型的标准差。

(3) 学生化残差

有时为了回避标准化残差的方差齐性假设，我们使用学生化残差，它和标准化残差的区别在于，其对每个 X 都单独计算标准误差。删除第 i 个样本点后，用剩余数据计算的回归系数为 $\hat{\beta}_{(i)}$ ，这时 σ^2 的估计值为

$$\hat{\sigma}_{(i)}^2 = \frac{1}{n - p - 2} \sum_{j \neq i} (Y_j - X_j \hat{\beta}_{(i)})^2$$

其中， X_j 是设计矩阵 X 的第 j 行， p 为自变量的个数， n 为样本容量。从而学生化残差等于

$$\frac{\hat{\varepsilon}_i}{\hat{\sigma}_{(i)} \sqrt{1 - h_{ii}}}$$

R 中用 `rstudent()` 计算标准化残差，调用格式与 `rstandard()` 类似，参数 `res` 指定模型残差。

```
rstudent(model, infl = lm.influence(model, do.coef = FALSE), res = infl$wt.res, ...)
```

9.3.2 影响分析

回归诊断要研究的另一个重要问题，是对参数估计或预测值有异常影响的数据，称为强影响数据。回归模型应当具有一定的稳定性，如果个别一两组数据对估计有异常大的影响，当我们剔除这些数据之后，将得到与原来差异很大的经验回归方程，从而我们将有理由怀疑原回归方程是否真正描述了变量之间的客观存在的相依关系。因此我们要考察每组数据对参数估计的影响大小，这部分内容在回归诊断中统称为影响分析。

(1) Leverage

Leverage 即帽子矩阵 H 的对角元素，由于 $\hat{Y} = HY$ ，从空间几何的角度， \hat{Y} 是将观测向量 Y 正交投影到由 X 的列向量所生成的子空间上的投影矩阵，有

$$\frac{\partial \hat{Y}_i}{\partial Y_i} = h_{ii}$$

h_{ii} 为 H 的对角元素，表示第 i 个样本对 \hat{Y}_i 的影响力。另外， \hat{Y}_i 的方差为 $Var(\hat{Y}_i) = h_{ii}\sigma^2$ ，所以 h_{ii} 也反映了 \hat{Y}_i 的波动情况。**Leverage** 的特点是只与预测量有关，由预测量的变差决定。如果 h_{ii} 较大，说明第 i 组样本的影响较大，是异常值点。Hoaglin 和 Welsch (1978) 给出的判断依据是

$$h_{ii} > \frac{2(p+1)}{n}$$

R 中计算 **Leverage** 的函数为 `hatvalues()` 和 `hat()`，调用格式为

```
hatvalues(model, infl = lm.influence(model, do.coef = FALSE), ...)
hat(x, intercept = TRUE)
```

其中的参数设置与残差函数类似，`model` 是回归分析 `lm()` 返回的对象；`x` 为设计矩阵。

```
> hii=hatvalues(lm.step) #计算Leverage
> p=4;n=20
> hii>2*(p+1)/n #将 hii 与经验判断标准进行比较
```

1	2	3	4	5	6	7	8	9	10	11	12	13
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
14	15	16	17	18	19	20						
FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE						

在财政收入的回归诊断中，有三个样本点的 **Leverage** 高于 $2(p+1)/n$ (判断结果为 **TRUE**)，说明可能是异常值点。

(2) DFFITS 统计量

另一种判断强影响点的方法是 DFFITS 统计量, 根据帽子矩阵的对角元素计算, 公式为

$$D_i(\sigma) = \sqrt{\frac{h_{ii}}{1-h_{ii}}} \cdot \frac{\hat{\varepsilon}_i}{\hat{\sigma}_{(i)}\sqrt{1-h_{ii}}}$$

对于第 i 个样本, 若

$$|D_i(\sigma)| > 2\sqrt{\frac{p+1}{n}}$$

则认为第 i 个样本是强影响点, 可能为异常值。R 中计算 DFFITS 统计量的函数是 `dffits()`, 调用格式为 `dffits(model, infl = , res =)`。对上例继续进行计算

```
> dff=dffits(lm.step)
> dff>2*sqrt((p+1)/n)
  1      2      3      4      5      6      7      8      9     10     11     12     13
TRUE  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
14     15     16     17     18     19     20
FALSE FALSE FALSE FALSE FALSE TRUE  FALSE
```

根据 DFFITS 准则, 发现第 19 个样本点的影响较大。

(3) Cook's 距离

Cook's 距离是 Cook 在 1977 年提出的一个统计量, 也用来发现对回归影响较大的数据元素, 定义为

$$D_i = \frac{(\hat{\beta} - \hat{\beta}_{(i)})^T X^T X (\hat{\beta} - \hat{\beta}_{(i)})}{(p+1)\hat{\sigma}^2}, \quad i = 1, 2, \dots, n$$

其中, $\hat{\beta}_{(i)}$ 为剔除第 i 个样本后由剩余样本拟合得到的回归系数。 D_i 越大的观测值越有可能是异常值点, 经验判断标准是, 当 $|D_i| > 4/n$ 时, 认为是强影响点。R 中用函数 `cooks.distance()` 计算 Cook's 距离。

```
> cook=cooks.distance(lm.step)
> cook>4/n
  1      2      3      4      5      6      7      8      9     10     11     12     13
TRUE  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
14     15     16     17     18     19     20
FALSE FALSE FALSE FALSE TRUE  TRUE  TRUE
```

根据 Cook's 距离判断的异常值点分别是第 1、18、19、20 个样本, 与 Leverage 和 DFFITS 统计量判断的强影响点一致。

(4) COVRATIO 准则

COVRATIO 顾名思义就是协方差的比值，用剔除第 i 个样本后计算的回归系数估计值的协方差矩阵，与完整样本估计得到的回归系数协方差矩阵作比较，两者分别为

$$Var(\hat{\beta}) = \sigma^2 (X^T X)^{-1}, \quad Var(\hat{\beta}_{(i)}) = \sigma^2 (X_{(i)}^T X_{(i)})^{-1}$$

COVRATIO 统计量为两个协方差行列式的比值

$$COVRATIO = \frac{\det(\hat{\sigma}_{(i)}^2 (X_{(i)}^T X_{(i)})^{-1})}{\det(\sigma^2 (X^T X)^{-1})} = \frac{(\hat{\sigma}_{(i)}^2)^{p+1}}{(\hat{\sigma}^2)^{p+1}} \frac{1}{1 - h_{ii}}$$

用样本点对应的 COVRATIO 值与 1 作比较，若差别很大，则说明该样本点对回归模型拟合效果的影响较大。R 中用函数 covratio() 计算 COVRATIO 统计量。

```
> covratio(lm.step)
      1      2      3      4      5      6      7      8      9     10     11
0.7419 1.2893 1.6064 0.3699 1.6074 0.7233 1.6636 1.6562 0.9875 1.5804 1.4751
     12     13     14     15     16     17     18     19     20
1.4620 1.5686 1.7414 1.7246 1.7229 1.6754 1.3233 0.4507 2.2776
```

以上 4 种统计量都是用于诊断异常值的方法，每种方法在 R 语言中都对应一个计算函数。另外，函数 influence.measures(model) 提供了这些影响分析的汇总概括，返回结果为一个列表，包括了 dfbeta、DFFITS、Cook 距离以及 COVRATIO 等多个统计量。

```
> options(digits=3) #打印结果显示 3 位小数
> influence.measures(lm.step)
```

```
Influence measures of
lm(formula = y ~ x3 + x4 + x5 + x6, data = revenue) :
      dfb.1_ dfb.x3 dfb.x4 dfb.x5 dfb.x6 dffit cov.r cook.d hat inf
1 -1.44679 -0.22591 1.119663 0.14732 1.32565 -2.1320 0.742 7.43e-01 0.510 *
2 0.38716 0.23223 -0.437053 -0.17219 0.36073 0.7525 1.289 1.11e-01 0.301
3 0.11252 0.08997 -0.112099 -0.07834 0.04643 0.2318 1.606 1.13e-02 0.190
4 0.37588 0.17465 -0.329567 -0.16019 -0.05680 0.8331 0.370 1.11e-01 0.126
5 -0.02465 -0.01482 0.037224 0.01097 -0.07884 -0.1292 1.607 3.55e-03 0.151
6 -0.14902 0.22479 0.088956 -0.20354 0.25529 -0.5588 0.723 5.71e-02 0.115
7 0.00169 -0.00975 -0.000987 0.00916 -0.00393 0.0135 1.664 3.89e-05 0.152
8 -0.00809 -0.03480 0.002729 0.03289 0.02379 0.0583 1.656 7.26e-04 0.153
9 0.13697 0.21365 -0.129581 -0.17927 -0.03701 -0.4043 0.987 3.19e-02 0.106
10 -0.01111 -0.00844 0.010843 0.00616 0.00185 0.0215 1.580 9.89e-05 0.108
11 0.16029 0.03967 -0.129195 -0.02057 -0.16064 -0.2749 1.475 1.58e-02 0.164
12 -0.14967 -0.00379 0.134607 -0.01589 0.08760 0.2153 1.462 9.73e-03 0.132
13 -0.06000 -0.01728 0.064140 0.00630 -0.01665 0.0906 1.569 1.75e-03 0.119
14 -0.02041 0.00180 0.016692 -0.00354 0.01936 0.0313 1.741 2.09e-04 0.190
15 -0.13676 -0.03369 0.212189 -0.00676 -0.35142 0.4415 1.725 4.05e-02 0.301
16 -0.06965 0.01742 0.102674 -0.03311 -0.14929 0.2116 1.723 9.49e-03 0.224
17 0.05366 -0.09005 -0.074534 0.09515 0.08105 -0.1885 1.675 7.53e-03 0.199
18 0.39012 -2.59618 -0.316284 2.46303 -0.82909 -2.9474 1.323 1.45e+00 0.689 *
19 0.68510 -0.12754 -0.641454 0.31319 -0.35434 1.5756 0.451 3.91e-01 0.329
20 -1.26681 1.17673 1.496385 -1.54518 -0.63371 -2.7384 2.278 1.35e+00 0.739 *
```

9.3.3 多重共线性诊断

多重共线性是指线性回归模型中的解释变量之间由于存在线性关系或近似线性关系，而使模型难以估计准确，这种现象在经济数据中尤为普遍。一般来说，产生多重共线性的原因主要有三个方面：

- 经济变量相关的共同趋势。例如经济繁荣期，各基本经济变量都趋于增长。
- 滞后变量的引入。例如消费受当期收入和前期收入的影响。
- 样本资料的限制。完全符合理论模型要求的样本数据很难收集，实际样本或多或少都会存在某种程度的共线性。

对于 $p(p > 2)$ 个自变量，如果存在常数 c_0, c_1, \dots, c_p 使得

$$c_1 X_1 + c_2 X_2 + \dots + c_p X_p = c_0$$

近似成立，则表示这 p 个变量间存在多重共线性。诊断多重共线性有以下几种方法。

(1) 特征根分析

若自变量之间存在共线性，则设计矩阵不是列满秩的，将 X_1, X_2, \dots, X_p 标准化和中心化后变换为主对角线是 1 的矩阵 X ，可以证明行列式 $|X^T X| \approx 0$ 。根据矩阵行列式的性质，行列式等于其特征根的连乘积，所以矩阵 $X^T X$ 至少有一个特征根近似为 0。若有 r 个特征根近似为 0，则回归设计矩阵 X 中有 r 个共线性关系，且共线性关系的系数向量 c_0, c_1, \dots, c_p 就是近似为 0 的特征根对应的特征向量。

R 中计算矩阵特征根和特征向量的函数是 `eigen()`，调用格式为

```
eigen(x, symmetric, only.values = FALSE, EISPACK = FALSE)
```

x 是待计算的矩阵；`symmetric` 若为 `TRUE`，则假设矩阵是对称的；`only.values` 默认返回特征根和特征向量，若为 `TRUE` 则仅返回特征根。使用函数 `eigen()` 特别要注意的一点是，矩阵 x 必须是经过标准化和中心化后的矩阵，其主对角线元素都是 1。在 R 软件中用函数 `cor()` 对矩阵计算相关系数其实就是对一个普通矩阵进行标准化的过程。

对财政收入案例中的 6 个变量样本组成的设计矩阵计算特征值：

```
> options(digits=3)
> xx=cor(revenue[3:8]) #提取设计矩阵并标准化
> eigen(xx)
$values
[1] 4.980418 0.838539 0.162066 0.012847 0.005649 0.000481

$vectors
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	-0.445	-0.0735	0.00746	0.8536	0.023853	-0.25950
[2,]	-0.444	-0.0582	0.26445	-0.0788	0.618251	0.58376
[3,]	-0.443	-0.0729	0.30977	-0.4538	0.137132	-0.69122
[4,]	-0.414	-0.1640	-0.86909	-0.2125	0.000528	0.03283
[5,]	-0.443	-0.0834	0.26783	-0.1168	-0.773369	0.33617
[6,]	0.201	-0.9757	0.08372	0.0200	0.017073	0.00212

最小的特征根为 0.000481, 十分接近 0, 说明 6 个自变量间一定存在多重共线性, 根据对应的特征向量[,6]还可以写出共线性关系。正是由于这种共线性的存在, 逐步回归中系统自动剔除了变量 x_1 、 x_2 。

(2) 条件数

特征根分析表明当矩阵 $X^T X$ 有一个特征根近似为零时, 设计矩阵 X 的列向量间必然存在共线性, 但这种共线性的严重程度如何呢? 为了更具体地分析, 我们引入一个重要指标——条件数, 它的计算公式为

$$\kappa(X^T X) = \frac{\lambda_{\max}(X^T X)}{\lambda_{\min}(X^T X)}$$

其中 λ_{\max} 、 λ_{\min} 分别表示方阵 $X^T X$ 的最大、最小特征根。直观上, 条件数刻画矩阵 $X^T X$ 特征根的最大差异, 特征根的最小值越接近于 0, 条件数的值越大, 多重共线性就越严重。经验上, $\kappa < 100$ 说明共线性程度较小; $100 \leq \kappa \leq 1000$ 说明变量间存在中等程度的共线性; 若 $\kappa > 1000$, 则认为存在严重的多重共线性。

R 软件提供了计算矩阵条件数的函数 `kappa()`, 其调用格式为

```
kappa(z, exact = FALSE, norm = NULL, method = c("qr", "direct"), ...)
```

z 为计算的矩阵; `exact` 表示逻辑值, 若为 `TRUE` 表示精确计算条件数, 默认为近似计算; `method` 指定使用的方法, 默认为 QR 分解。

```
> kappa(xx)
[1] 6132
```

在财政收入的例子中, 包含所有变量样本数据的设计矩阵条件数是 6132 > 1000, 故认为多重共线性十分严重。

(3) 方差扩大因子 (VIF)

统计上可以证明, 自变量 X_j 的回归系数估计值 $\hat{\beta}_j$ 的方差可以表示为

$$\text{Var}(\hat{\beta}_j) = \frac{\sigma^2}{\sum x_j^2} \cdot \frac{1}{1 - R_j^2} = \frac{\sigma^2}{\sum x_j^2} \cdot \text{VIF}_j$$

其中, VIF_j 即为变量 X_j 的方差扩大因子, R_j^2 为自变量 X_j 对其余自变量作回归分析的复相关系数。方差扩大因子越大, 表明自变量之间的多重共线性越严重, 造成回归系数的估计值很不稳定; 反之, VIF 越接近于 1, 共线性越弱。经验表明, $VIF > 10$ 说明模型中有很强的共线性。

R 软件的统计包 DAAG 中, 包含有计算方差扩大因子的函数 `vif()`, 其调用格式为 `vif(obj, digits=5)`。`obj` 表示用函数 `lm()` 生成的回归分析对象; `digits` 给出数字的小数点位数, 默认为 5。另外, 程序包 `fmsb` 的函数 `VIF()` 也可以计算方差扩大因子。例如, 在 R 中输入指令

```
> lm.reg=lm(y~x1+x2+x3+x4+x5+x6,data=revenue)
> library(DAAG)
> vif(lm.reg,digits=3)
      x1      x2      x3      x4      x5      x6
197.00  778.00 1010.00  10.50  343.00   1.28
```

计算结果显示, 除了 X_6 以外所有变量的方差扩大因子均大于 10, 说明模型中存在很强的多重共线性。使用方差扩大因子 VIF 判断共线性还有一个好处是, 它可以初步判断哪些变量之间存在共线性。例如, 变量 X_2 和 X_3 方差扩大因子最大, 初步判断它们之间可能存在很强的相关性, 计算它们的相关系数:

```
> attach(revenue)
> cor(x2,x3)
[1] 0.998
```

可知, X_2 和 X_3 的线性相关系数高达 0.998, 因此可以肯定它们之间存在严重的共线性, 所以上面作逐步回归的过程中就剔除了变量 X_2 。

9.4 岭回归及 R 实现

在多元线性回归分析中, 我们会在众多变量中选择对因变量显著性影响大的那些自变量。但这时我们常常会遇到一个问题: 在某些情况下, 增加或剔除一个自变量后回归系数变化很大, 甚至改变符号。为什么会出现这种情况呢? 主要的原因是变量之间存在多重共线性, 因此我们要引入岭回归。

岭回归分析是一种专用于共线性数据分析的有偏估计回归方法, 实质上是一种改良的最小二乘法, 它是通过放弃最小二乘法的无偏性, 以损失部分信息、降低精度为代价获得回归系数更为符合实际、更可靠的回归方法, 对病态数据的耐受性远远强于最小二乘法。

岭回归的想法比较直观: 当自变量间存在多重共线性时, 有 $|X'X| \approx 0$, 我们设想给 $X'X$ 加上一个正常数矩阵 $kI (k > 0)$, 那么 $X'X + kI$ 接近奇异的程度就会比 $X'X$ 小很多, 从而消除了多重共线性。

而考虑到变量的量纲, 应首先对数据进行标准化, 为简便计算, 标准化后的设计矩阵仍用 X

表示。我们称

$$\hat{\beta}(k) = (X'X + kI)^{-1} X'y$$

为 β 的岭回归估计，其中 k 为岭参数。显然，岭回归估计 β 值比最小二乘估计值稳定，当 $k=0$ 时的岭回归估计就是普通最小二乘估计。

由于岭参数 k 不是唯一确定的，所以我们得到的岭回归估计 $\hat{\beta}(k)$ 实际是回归参数 β 的一个估计族。当岭参数 k 在 $(0, \infty)$ 内变动时， $\hat{\beta}_j(k)$ 是 k 的函数，在平面坐标系上把函数 $\hat{\beta}_j(k)$ 描画出来，形成的曲线称为岭迹，岭迹图如图 9.7 所示。

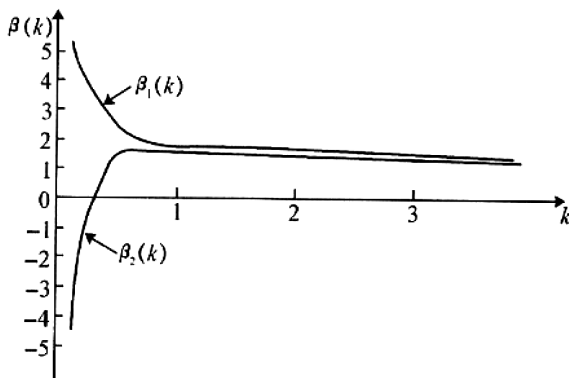


图 9.7 岭迹图

根据岭迹图我们可以选择合适的 k 值，称为岭迹法，其一般原则是：

- (1) 各回归系数的岭估计基本稳定；
- (2) 最小二乘估计的回归系数符号不合理时，岭估计参数的符号变得合理；
- (3) 回归系数没有不合乎实际意义的绝对值；
- (4) 残差平方和增加不太多。

R 的核心程序包 MASS 中有专门用于岭回归分析的函数 `lm.ridge()`，其调用格式为

```
lm.ridge(formula, data, subset, na.action, lambda = 0, model = FALSE,
x = FALSE, y = FALSE, contrasts = NULL, ...)
```

其中，`formula` 是回归模型公式表达形式，形如 `response ~ predictors`；`data` 指定数据的数据框；当只需要 `data` 的一个子集参与计算时，用参数 `subset` 来设置；`na.action` 表示遇到缺失值时应采取的行为；`lambda` 是岭参数的标量或矢量；`model`、`x` 和 `y` 均为逻辑值，分别表示结果是否返回模型框架、设计矩阵和响应变量。

以商品销售量为相应变量 Y ，考虑有关的 4 个因素： X_1 ——可支配收入； X_2 ——该商品平均价格指数； X_3 ——该商品的社会保有量； X_4 ——其他消费品平均价格指数。数据如表 9.6 所示。

表 9.6 商品销售量及相关因素的数据

商品销售量 Y	可支配收入 X_1	平均价格指数 X_2	社会保有量 X_3	其他消费品平均价格指数 X_4
8.4	82.9	92	17.1	94
9.6	88	93	21.3	96
10.4	99.9	96	25.1	97
11.4	105.3	94	29	97
12.2	117.7	100	34	100
14.2	131	101	40	101
15.8	148.2	105	44	104
17.9	161.8	112	49	109
19.6	174.2	112	51	111
20.8	184.7	112	53	111

首先输入数据并查看数据的多重共线性情况，计算 $X'X$ 的条件数。在 R 中输入如下指令。

```
> y=c(8.4,9.6,10.4,11.4,12.2,14.2,15.8,17.9,19.6,20.8)
> x1=c(82.9,88,99.9,105.3,117.7,131,148.2,161.8,174.2,184.7)
> x2=c(92,93,96,94,100,101,105,112,112,112)
> x3=c(17.1,21.3,25.1,29,34,40,44,49,51,53)
> x4=c(94,96,97,97,100,101,104,109,111,111)
> x=cbind(x1,x2,x3,x4) #将数据按列合并
> xx=crossprod(x) #计算矩阵交叉积，结果为矩阵  $X'X$ 
> kappa(xx,exact=T) #计算条件数
[1] 81629.86
```

得到的条件数 $K=81629.86 > 1000$ ，认为有严重的多重共线性。因此，不能直接用最小二乘法估计拟合回归方程。考虑用岭回归估计方法分析变量之间的关系，首先绘制岭迹图：

```
> library(MASS)
> plot(lm.ridge(y~x1+x2+x3+x4,lambda=seq(0,0.5,0.001)))
```

绘制结果如图 9.8 所示。

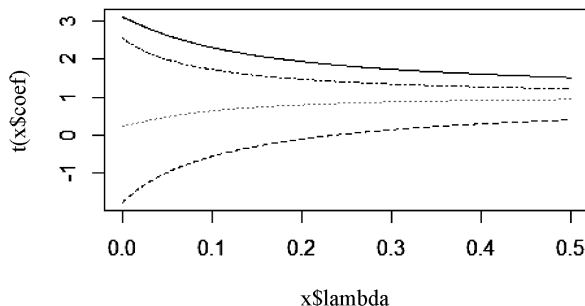


图 9.8 商品销售量的岭迹图

从图中可以看出，曲线变平稳的速度很慢，很难直接得出适当的岭参数 k 值，而 R 可以通过函数 `select()` 计算出根据几个统计量得到的 k 值：

```
> select(lm.ridge(y~x1+x2+x3+x4,lambda=seq(0,0.5,0.001)))
modified HKB estimator is 0.004265631
modified L-W estimator is 0.004895202
smallest value of GCV at 0.009
```

其中，HKB 和 L-W 分别为不同方法下计算得到的岭参数估计值，据此我们选择 $\lambda=0.0045$ ，从而给出相应的参数估计：

```
> options(digits=3)
> lm.ridge(y~x1+x2+x3+x4,lambda=0.0045)
           x1      x2      x3      x4
-17.5554   0.0887  -0.2180   0.0202   0.4073
```

最终得到的岭回归方程为

$$y = -17.5554 + 0.0887x_1 - 0.218x_2 + 0.0202x_3 + 0.4073x_4$$

9.5 广义线性模型

9.5.1 模型理论

广义线性模型（Generalized Linear Model）是一般线性模型的推广，由 Nelder 和 Wedderburn(1972)首次提出。它使因变量的总体均值通过一个非线性连接函数而依赖于线性预测值，允许响应概率分布为指数分布族中的任何一员。许多广泛应用的统计模型都属于广义线性模型，如常用于研究二元分类响应变量的 Logistic 回归、Poisson 回归和负二项回归模型等。一个广义线性模型包含以下三个部分：

- ① 随机成分。观察值 Y_i 相互独立，且服从指数分布族。
- ② 线性成分。 $\eta_i = \beta_0 + X_{i1}\beta_1 + \dots + X_{ip}\beta_p$ 。
- ③ 连接函数 g 。 g 单调可微， $\eta_i = g(\mu_i)$ 。

因此广义线性模型的一般形式是

$$g(\mu_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} + \varepsilon_i$$

Y_i 服从指数分布族，其概率密度可表示为

$$f(y_i, \theta_i, \varphi) = \exp \left(\frac{y_i \cdot \theta_i - b(\theta_i)}{a(\varphi)} + c(y_i, \varphi) \right)$$

其中, θ_i 称为自然参数, 它与均值有关, 对每个观察值是不同的; ϕ 为离散参数, 与方差有关, 对每个观察值都是相同的; a 、 b 、 c 为已知函数, 对所有观察值具有相同形式。因此, 每个观察值 Y_i 都来自于同一个分布, 但它们的均值因自然参数 θ_i 的不同而变化。密度函数可以表示为上述形式的分布, 包括正态分布、泊松分布、伽马分布、二项分布、逆高斯分布等, 如表 9.7 所示。

表 9.7 各种常见的指数型分布及其主要参数

分 布	θ	$b(\theta)$	ϕ	$E(y) = b'(\theta)$	$Var(y) = b''(\theta)\phi$
正态分布	μ	$\theta^2 / 2$	σ^2	$\mu = \theta$	σ^2
逆高斯分布	$\frac{1}{\mu^2}$	$-(-2\theta)^{1/2}$	σ^2	$\mu = \frac{1}{\sqrt{\theta}}$	$\mu^3 \sigma^2$
伽玛分布	$\frac{1}{\mu}$	$-\ln(\theta)$	$\frac{1}{\gamma}$	$\mu = \frac{1}{\theta}$	$\mu^3 \gamma$
二项分布	$\ln \frac{p}{1-p}$	$\ln(1 + e^\theta)$	1	$p = \frac{e^\theta}{1 + e^\theta}$	$p(1-p)$
泊松分布	$\ln \lambda$	e^θ	1	$\lambda = e^\theta$	λ
负二项分布	$\ln \lambda$	e^θ	k	$\lambda = e^\theta$	$\lambda + k\lambda^2$

典型的连接函数有 4 种, 总结在表 9.8 中。

表 9.8 典型的连接函数及对应分布

变换	连接函数	回归模型	常见对应分布
恒等	$\mu_i = \eta_i$	$E(y) = x^T \beta$	正态分布
对数	$\log(\mu_i) = \eta_i$	$E(y) = \exp(x^T \beta)$	泊松分布
logit	$\log[\mu_i / (1 - \mu_i)] = \eta_i$	$E(y) = \frac{\exp(x^T \beta)}{1 + \exp(x^T \beta)}$	二项分布
逆	$1/\mu_i = \eta_i$	$E(y) = \frac{1}{x^T \beta}$	伽马分布

广义线性模型的参数估计一般不能用最小二乘估计, 常用加权最小二乘法或最大似然法估计, 各回归系数 β 需用迭代方法求解。

9.5.2 R 语言实现

R 提供了拟合广义线性模型的函数 `glm()`, 其调用格式为

```
glm(formula, family = gaussian, data, weights, subset, na.action, start = NULL,
    etastart, mustart, offset, control = list(...), ...)
```

其中, `formula` 为拟合公式, 与函数 `lm()` 中的参数 `formula` 用法相同; 最重要的参数是 `family`, 用于指定分布族, 包括正态分布 (`gaussian`)、二项分布 (`binomial`)、泊松分布 (`poisson`) 和伽马分

布(Gamma), 分布族还可以通过选项 `link` 来指定连接函数, 默认值为 `family=gaussian (link=identity)`, 而二项分布默认的连接函数是 `logit`, 即 `family=binomial(link=logit)`; `data` 指定数据集; `offset` 指定线性函数的常数部分, 通常反映已知信息; `control` 用于对待估参数的范围进行设置。

普通线性模型要求变量必须是连续的, 但广义线性模型可以很好地处理分类变量。例如保险数据中的很多变量都是分类的定性变量, 在 R 中对其建立广义线性模型可以较好地解释变量关系。对某车险保单, 考虑如下的保单索赔次数数据, 根据车型、被保险人性别等因素对保单进行分组 (见表 9.9), 考察这两个因素对索赔次数的影响。

表 9.9 车险保单索赔次数分组数据

车型	性别	风险暴露数	索赔次数
小	男	500	42
中	男	1200	37
大	男	100	10
小	女	400	101
中	女	500	73
大	女	300	14

已知索赔次数服从泊松分布, 相应的连接函数常用对数连接函数, 模型可以写为

$$\mu_i = E(y_i) = e_i \cdot \exp(x_{i1}\beta_1 + x_{i2}\beta_2 + \dots + x_{ip}\beta_p)$$

其中, y 是因变量索赔次数; e_i 是风险单位数, 其对数值属于模型线性部分的常数项, 应当用参数 `offset` 指定; X 为解释变量——包含车型和性别取值的矩阵。

下面用 R 实现, 首先建立数据集, 分类变量直接输入定性的取值即可, `glm()` 分析时会自动转换成矩阵 X , 注意参数 `family` 的写法。

```
> dat=data.frame(y=c(42, 37, 10, 101, 73, 14),n=c(500, 1200, 100, 400, 500, 300),
+ type=rep(c('小','中','大'),2),gender=rep(c('男','女'),each=3))
> dat$logn=log(dat$n) #风险暴露数取对数
> dat.glm=glm(y~type+gender,offset=logn,data=dat,family=poisson(link=log))
#offset 风险单位数事先已知
> summary(dat.glm) #glm的输出结果

Call:
glm(formula = y ~ type + gender, family = poisson(link = log),
    data = dat, offset = logn)

Deviance Residuals:
    1     2     3     4     5     6 
0.333 -1.498  3.792 -0.209  1.212 -1.780
```

```

Coefficients:
      Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.797      0.236  -16.06 < 2e-16 ***
type 小       1.268      0.222   5.71 1.1e-08 ***
type 中       0.554      0.230   2.42  0.016 *
gender 女     1.173      0.132   8.91 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 163.490 on 5 degrees of freedom
Residual deviance: 21.413 on 2 degrees of freedom
AIC: 61.68

Number of Fisher Scoring iterations: 5

```

表 9.5 列举的提取函数对 `glm()` 的返回对象同样适用。在 R 中建立广义线性模型的方法很简单，`summary()` 输出的结果包含很多信息。在本例中，估计的回归系数都是非常显著的；Null deviance 可以认为是模型的残差，它的值越小说明模型拟合效果越好；模型的 AIC 统计量为 61.68，它和 deviance 一起可以用来作为判断标准，选取合适的分布族和链接函数。

下面通过作图来观察模型拟合的效果，首先提取模型的预测值，注意函数 `predict()` 提取的是线性部分的拟合值，在对数连接函数下，要得到 Y 的拟合值，应当再做一次指数变换。以实际观测值为横坐标，模型拟合值为纵坐标作图，散点越接近直线 $y=x$ ，说明模型的拟合效果越好。

```

> dat.pre=predict(dat.glm)
> layout(1) #取消绘图区域分割
> plot(y,exp(dat.pre),xlab='观测值',ylab='拟合值',main="索赔次数的拟合效果",pch="*")
> abline(0,1) #添加直线 y=x, 截距为 0, 斜率为 1

```

绘制结果如图 9.9 所示。

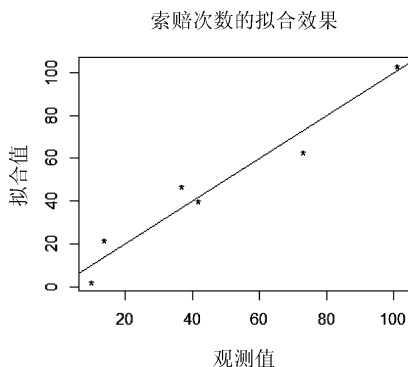


图 9.9 泊松回归拟合效果图

如果自变量很多, 最终的模型中很可能包含不显著的变量, 类似于普通线性模型, 这时可以用 `step()` 做变量筛选。

除了上面提到的 4 种典型分布, 另一个常用的分布族是负二项分布, 函数 `glm()` 中的参数 `family` 不能直接指定, 这时要使用程序包 `MASS` 中的函数 `glm.nb()` 建立负二项回归, 注意 `offset` 的写法稍有不同。若假设上例中的索赔次数服从负二项分布, 在 R 中应输入指令:

```
> library(MASS)
> attach(dat)
> dat.glmnb=glm.nb(y~type+gender+offset(logn)) #负二项回归
Warning message:
In glm.nb(y ~ type + gender + offset(logn)) : alternation limit reached
> summary(dat.glmnb) #输出结果

Call:
glm.nb(formula = y ~ type + gender + offset(logn), init.theta = 6.879011721,
link = log)

Deviance Residuals:
    1      2      3      4      5      6
-0.395 -1.020  1.535  0.350  0.810 -1.646

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.0541    0.4172  -7.32  2.5e-13 ***
type 小      0.7430    0.4494   1.65  0.098 .
type 中      0.0208    0.4514   0.05  0.963
gender 女     0.7996    0.3528   2.27  0.023 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(6.88) family taken to be 1)

Null deviance: 16.6831 on 5 degrees of freedom
Residual deviance: 7.0412 on 2 degrees of freedom
AIC: 60.45

Number of Fisher Scoring iterations: 1

Theta: 6.88
Std. Err.: 5.24
Warning while fitting theta: alternation limit reached

2 x log-likelihood: -50.45
```

负二项回归拟合的模型 AIC 为 60.45, 残差 Null deviance 为 16.6831, 小于泊松回归拟合的残差值, 说明负二项分布的广义线性模型更加稳定, 但从回归系数的显著性上看, 泊松回归拟合的变量系数更加显著。

第 10 章

主成分分析与因子分析

在数据分析中，我们常常面对判断某一事物在同类事物中的好坏、优劣程度及其发展规律等问题，然而影响某一事物的特征及发展规律的因素是多元化的，为了更加深入地分析，我们需要综合与其相关的各种影响因素进行综合分析和评价。但在上一章我们曾提出，多变量、大样本的数据会带来多重共线性等问题，各影响因素所反映的信息重复，反而会影响统计结果的真实性和科学性。因此，为了尽量避免信息重叠和减轻工作量，人们提出了“降维”的思想，找出少数几个互不相关的综合变量来尽可能地反映原来数据所含有的绝大部分信息，主成分分析和因子分析正是为解决此类问题而产生的多元统计分析方法。

这两种方法有相似的功能，出发点都是变量的相关系数矩阵，目的在于寻找多个变量的“代表”，所以常常被一起提及。但它们之间既有联系又有区别，我们在实际使用中应当透彻地理解两种方法，避免混淆。本章我们将介绍两种方法的模型理论，并用实际案例具体说明如何用 R 软件进行主成分分析和因子分析。

10.1 主成分分析

10.1.1 理论基础

主成分分析试图在保证数据信息丢失最少的原则下，将多变量的截面数据集进行最佳综合简化，简单地说就是根据多个指标之间的联系，选出它们的某种线性组合，从而化为少数几个综合指标。

主成分的概念最早由 Karl parson 在 1901 年引进，但当时只是对非随机变量讨论的，1933 年 Hotelling 将这个概念推广到随机变量。一个十分著名的案例是美国统计学家 Stone 在 1947 年关于国民经济的研究，他使用美国 1929–1938 年的数据，总结了可以反映国民收入与支出情况的 17

个变量，包括雇主补贴、消费资料和生产资料、纯公共支出、股息、利息，等等。在进行主成分分析后，他仅用三个新的指标就取代了原来的 17 个变量，解释度高达 97.4%。根据这些变量之间的联系和经济学知识，Stone 给这三个新指标分别命名为总收入 F_1 、总收入变化率 F_2 和经济发展趋势 F_3 。

主成分是由原始指标综合形成的几个新指标，即上例中的 F_1 、 F_2 和 F_3 。根据主成分所含信息量的大小称为“第一主成分”、“第二主成分”等。主成分与原始变量之间的关系为：

- ① 主成分保留了原始变量绝大多数信息；
- ② 主成分个数远远少于原始变量的个数；
- ③ 各个主成分之间互不相关；
- ④ 每个主成分都是原始变量的线性组合。

下面以具体的数学模型来说明主成分分析的原理。

(1) 主成分的数学模型

设 X_1, X_2, \dots, X_p 为 p 维随机变量，主成分分析就是将 p 个观测变量通过线性组合转化成为 p 个新的指标，即

$$\begin{aligned} F_1 &= u_{11}X_1 + u_{12}X_2 + \cdots + u_{1p}X_p \\ F_2 &= u_{21}X_1 + u_{22}X_2 + \cdots + u_{2p}X_p \\ &\vdots \\ F_p &= u_{p1}X_1 + u_{p2}X_2 + \cdots + u_{pp}X_p \end{aligned}$$

模型满足如下的条件：

- ① 每个主成分的系数平方和为 1，即 $u_{i1}^2 + u_{i2}^2 + \cdots + u_{ip}^2 = 1$ ；
- ② 主成分之间相互独立，无重叠的信息，即

$$\text{Cov}(F_i, F_j) = 0, \quad i \neq j, \quad i, j = 1, 2, \dots, p$$

- ③ 主成分的方差依次递减，重要性依次递减

$$\text{Var}(F_1) \geq \text{Var}(F_2) \geq \cdots \geq \text{Var}(F_p)$$

主成分分析的目的是简化变量，主成分的个数通常小于原始变量的个数，但实际问题中具体应该保留几个主成分呢？我们自然希望主成分尽可能多地反映原来变量的信息，所以要权衡主成分个数和保留的信息，这里的“信息”用方差来测量，即 $\text{Var}(F_1)$ 越大表示 F_1 包含的信息越多。最终，新的指标 $F_1, F_2, \dots, F_k (k \leq p)$ 按照保留主要信息量的原则充分反映原指标的信息，并且相互独立。

(2) 主成分的导出

以上是主成分分析的原理, 接下来我们在数学模型的基础上, 根据原始数据和模型的三个条件, 导出主成分。主成分的数学模型写为矩阵形式是

$$F = \begin{pmatrix} u_{11} & u_{12} \cdots & u_{1p} \\ u_{21} & u_{22} \cdots & u_{2p} \\ \vdots & \vdots & \vdots \\ u_{p1} & u_{p2} \cdots & u_{pp} \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{pmatrix} = AX$$

模型要求主成分之间互不相关, 所以主成分之间的协方差阵应该是一个对角阵。

$$\text{Var}(F) = \text{Var}(AX) = A \text{Var}(X) A^T = A \Sigma_X A^T = \Lambda = \begin{pmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \lambda_p \end{pmatrix}$$

若分析中所选择的经济变量具有不同的量纲, 变量水平差异很大, 则容易造成不合理的结果。为了避免量纲的影响, 我们常将随机变量标准化, 标准化后的协方差阵就是 X 的相关系数阵, 所以主成分分析应该基于相关系数矩阵。设相关阵为 R , 即有 $R = XX^T$, 用 R 代替 Σ_X

$$\text{Var}(F) = AXX^T A^T = ARA^T = \Lambda, \quad RA^T = A^T \Lambda$$

Σ_X 的特征根 $\lambda_1, \lambda_2, \dots, \lambda_p$ 分别代表主成分 F_1, F_2, \dots, F_p 的方差, 且 $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$, A^T 的列向量是 λ_i 对应的特征向量。因此主成分的导出转化为计算相关系数矩阵的特征值 λ_i , $u_i = (u_{i1}, u_{i2}, \dots, u_{ip})$ 则是相应的特征向量。

$$|R - \lambda_i I| = 0$$

最后一个步骤是根据方差贡献率选择重要的主成分, 并写出主成分表达式。

通过计算特征向量可以得到 p 个主成分, 其方差是递减的, 因此包含的信息量也递减。为了实现降维, 我们要根据各个主成分累计贡献率的大小选取前 k 个主成分。贡献率指某个主成分的方差占全部方差的比重, 也就是某个特征值占全部特征值合计的比重。

$$\text{贡献率} = \frac{\lambda_j}{\sum_{i=1}^p \lambda_i}, \quad j = 1, 2, \dots, p$$

贡献率越大, 说明该主成分所包含的原始变量的信息越强。主成分个数 k 的选取, 主要根据主成分的累计方差贡献率来决定, 一般要求累计贡献率达到 80%~85% 以上, 这样才能保证新的综

合变量能包括原始变量的绝大多数信息。

碎石图 (Scree plot) 也可以作为判断主成分个数的标准, 如图 10.1 所示。它以成分数为横坐标, 特征值为纵坐标。碎石图相当于特征值变化趋势图, 特征值由陡峭变为平坦的转折点即为选择主成分的最佳个数, 例如图 10.1 中我们可以选择 4 个主成分, 更精确一点, 可以选择 6 个主成分。

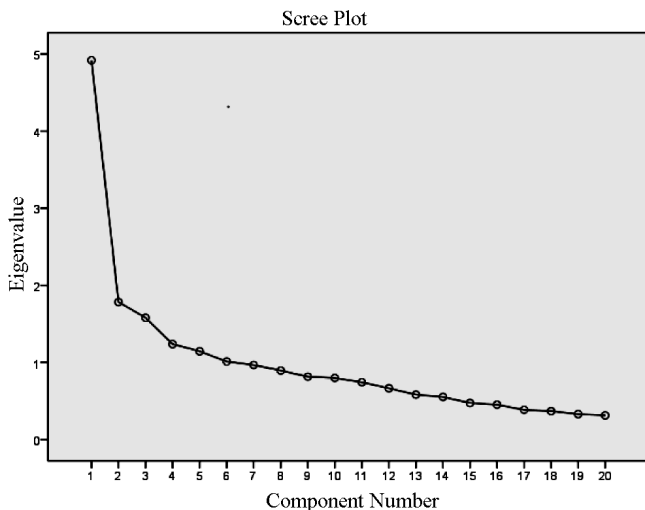


图 10.1 碎石图

F_j 与 X 的第 i 个分类的相关系数称为因子载荷量, 即

$$\rho(X_i, F_j) = \frac{u_{ij}\lambda_j}{\sqrt{\sigma_{ii}}\sqrt{\lambda_j}} = \frac{u_{ij}\sqrt{\lambda_j}}{\sqrt{\sigma_{ii}}}$$

因子载荷的平方可看作为第 j 个主成分可解释 X_j 多少比率的信息, 说明 X_j 在第 j 个主成分中的相对重要性。

(3) 主成分分析的步骤

- ① 计算相关系数矩阵;
- ② 计算相关系数矩阵的特征根及对应的特征向量;
- ③ 选出最大的特征根, 对应的特征向量等于第一主成分的系数;
选出第二大的特征根, 对应的特征向量等于第二主成分的系数;
.....
- ④ 计算累积贡献率, 选择恰当的主成分个数;
- ⑤ 写出前 k 个主成分的表达式。

在主成分分析中还有一个非常关键的问题是如何给主成分赋予新的、合理的解释意义，这是根据主成分表达式的系数结合理论知识来进行的。主成分是原来变量的线性组合，主要综合系数绝对值大的那几个变量，若几个变量系数大小相当，应认为这一主成分是这几个变量的总和。但应该赋予怎样的解释意义，还要结合具体实际问题 and 专业知识，进而才能达到深刻分析的目的。

10.1.2 R 语言实现

R 语言中，有两个进行主成分分析的函数，首先是 `prcomp()`，其调用格式有两种：

```
princomp(formula, data = NULL, subset, na.action, ...)
```

其中，`formula` 类似于 `lm()` 中的参数，用于指定模型表达式，但主成分分析中没有响应变量；`data` 指定数据框；`subset` 用于选择数据矩阵的行，选出数据的一个子集进行分析；`na.action` 表示数据包含缺失值时应该采取什么措施。

```
princomp(x, cor = FALSE, scores = TRUE, covmat = NULL,
         subset = rep(TRUE, nrow(as.matrix(x))), ...)
```

`x` 是用于主成分分析的数据集；`cor` 默认为 `FALSE`，表示使用样本的协方差阵作主成分分析，若 `cor=TRUE` 则使用相关系数矩阵 `R` 求主成分；参考“主成分导出”的数学推导，使用相关系数矩阵，其实就相当于将数据进行标准化再计算主成分。如果数据不用参数 `x` 指定，可以用 `covmat` 直接指定协方差阵。其他参数的意义参考 R 帮助。

查看 `princomp()` 的源代码，可以发现它是基于特征向量 `eigen()` 函数计算的，有可能会计算出数值绝对值正确但符号错误的问题。而 R 中的另一个主成分分析函数是 `prcomp()`，它的计算方法基于矩阵的 SVD 奇异值分解 (`svd()` 函数)，相比 `princomp()` 在数学性质上更稳定。它的第一种调用方式与 `princomp()` 一致，另一种为

```
prcomp(x, retx = TRUE, center = TRUE, scale. = FALSE, tol = NULL, ...)
```

`x` 是用于主成分分析的数据集；`retx` 是逻辑值，指示是否旋转数据，即中心化和标准化。上文我们讲到，实际分析中为了避免量纲对方差的影响，通常在使用中我们都要将数据标准化，所以这一参数的默认值为 `TRUE`；`center` 也是逻辑值，指示变量是否中心化；若 `scale.=TRUE` 即使用相关阵计算主成分，默认使用协方差矩阵。

利用主成分分析函数返回一个对象后，通过提取函数（上一章曾介绍过）可以得到不同的信息。常用的函数 `loadings()` 是“载荷”的意思，所以用来显示主成分分析或因因子分析中的载荷系数，在主成分分析中，即主成分对应原始变量线性组合的系数。用函数 `summary()` 提取信息，其中有一个参数 `loadings` 是逻辑值，若 `loadings=TRUE` 直接显示载荷系数，从而不需要再用函数 `loadings()` 提取。类似于回归分析，还可以用函数 `predict()` 预测主成分的值。

随着 R 程序包的开发，多元数据分析程序包 `labdsv` 中也有专门进行主成分分析的函数 `pca()`，`pca` 即主成分分析 `principal components analysis` 的缩写。这个版本是一个简单的 `prcomp` 功能，所

以在使用上与 `prcomp()` 基本一致，调用格式为

```
pca(mat, cor = FALSE, dim = min(nrow(mat), ncol(mat)))
```

`mat` 表示矩阵或数据框，每行为一个样本；`cor` 表示是否使用相关系数矩阵计算，默认为 `FALSE`，表示使用协方差阵计算，但实际分析中为了消除量纲对方差的影响，我们通常要设置 `cor=TRUE`；`dim` 为返回的维数，即提取主成分的个数。

利用 `pca()` 得到主成分分析的结果后，返回一个对象存储分析结果，我们可以继续使用程序包 `labdsv` 中的函数 `varplot.pca()` 来绘制碎石图以及累计方差图；函数 `loadings.pca()` 提取载荷系数。

主成分分析作为一种常用的多变量分析方法，其实际应用十分广泛，如人口统计学、数量地理学、经济分析等。接下来以某地区农业生态经济系统各区域单元相关指标数据为案例，在 R 软件中运用主成分分析方法，综合更少的指标信息描述该地区农业生态经济的发展状况。部分数据如表 10.1 所示。

表 10.1 某农业生态经济系统各区域单元的有关数据

样本 序号	X_1 人口密度 (人/km ²)	X_2 人均耕地 面积(ha)	X_3 森林覆 盖率(%)	X_4 农民人均纯 收入(元/人)	X_5 人均粮食产 量(kg/人)	X_6 经济作物占 农作物播面 比例(%)	X_7 耕地占土地 面积比率 (%)	X_8 果园与林地 面积之比 (%)	X_9 灌溉田占耕 地面积之比 (%)
1	363.912	0.352	16.101	192.11	295.34	26.724	18.492	2.231	26.262
2	141.503	1.684	24.301	1752.35	452.26	32.314	14.464	1.455	27.066
3	100.695	1.067	65.601	1181.54	270.12	18.266	0.162	7.474	12.489
4	143.739	1.336	33.205	1436.12	354.26	17.486	11.805	1.892	17.534
5	131.412	1.623	16.607	1405.09	586.59	40.683	14.401	0.303	22.932
...

在 R 中首先读取文本文档数据，读入后形成数据框对象，可以直接对其进行主成分分析。注意，通过参数设置我们可以选择直接使用相关系数矩阵计算主成分，从而剔除量纲对方差计算的影响，这相当于将数据标准化，所以读入数据后不需要做事先的标准化处理。

```
> agri=read.table("d:/data/agriculture.txt",header=TRUE)
> agri=agri[,-1] #剔除第一列序号
> agri.pr=princomp(agri,cor=TRUE) #cor=TRUE 表示用相关阵计算
> options(digits=4) #结果显示 4 位有效数字
> summary(agri.pr,loadings=TRUE) #loadings=TURE 选项列出主成分对应原始变量的系数
```

Importance of components:

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5	Comp.6	Comp.7
Standard deviation	2.1590	1.4455	1.0213	0.71234	0.56140	0.4389	0.33821
Proportion of Variance	0.5179	0.2322	0.1159	0.05638	0.03502	0.0214	0.01271

Cumulative Proportion	0.5179	0.7501	0.8660	0.92234	0.95736	0.9788	0.99147
	Comp.8		Comp.9				
Standard deviation	0.212900		0.177407				
Proportion of Variance	0.005036		0.003497				
Cumulative Proportion	0.996503		1.000000				

Loadings:

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5	Comp.6	Comp.7	Comp.8	Comp.9
x1	0.342	-0.368		-0.375	-0.355	0.312	0.559	0.113	0.233
x2		0.614		0.155	-0.761	-0.110			
x3	-0.44					0.206	0.467	-0.203	-0.692
x4		0.601		-0.598	0.310	0.395			0.139
x5	0.376	0.307			0.396	-0.508	0.580		
x6	0.379	0.124	0.122	0.620	0.154	0.638			
x7	0.432		-0.246	-0.148			-0.241	-0.777	-0.235
x8			0.950					-0.231	
x9	0.446			-0.224		-0.136	-0.246	0.532	-0.613

分析结果的 **Standard deviation** 表示主成分的标准差, 即相应特征值 (等于方差) 的平方根; **Proportion of Variance** 表示方差贡献率, **Cumulative Proportion** 表示累计方差贡献率。第一主成分 F_1 的贡献率为 51.79%, 第二主成分 F_2 的贡献率为 23.22%, 第三主成分 F_3 的贡献率为 11.59%, 前三个主成分的累计方差贡献率达到 86.6%, 因此最终选取 3 个主成分即可, 其余 6 个舍去。写出主成分与原变量的线性关系式为

$$F_1 = 0.342X_1 - 0.368X_2 - 0.375X_4 - 0.355X_5 + 0.312X_6 + 0.599X_7 + 0.113X_8 - 0.233X_9$$

$$F_2 = 0.614X_2 + 0.155X_4 - 0.761X_5 - 0.11X_6$$

$$F_3 = -0.446X_2 + 0.206X_6 + 0.467X_7 - 0.203X_8 + 0.692X_9$$

最后, 根据主成分与原始变量之间的线性关系, 给出各成分的实际解释意义。

① 第一主成分 F_1 是多个变量的线性组合, 且各变量的系数大小相当, 与 X_1 、 X_6 、 X_7 呈现较强的正相关, 与 X_2 、 X_4 、 X_5 、 X_9 呈现较强的负相关, 这几个变量综合反映了生态经济结构状况, 因此我们认为第一主成分 F_1 是生态经济结构的总体代表。

② 第二主成分 F_2 与 X_2 、 X_4 正相关, 与 X_5 呈现较强的负相关, X_2 、 X_4 、 X_5 分别为人均耕地面积、人均纯收入和人均粮食产量, 都反映了人均占有资源的情况, 因此认为第二主成分 F_2 代表了人均资源量。

③ 第三主成分 F_3 与 X_2 (人均耕地面积)、 X_8 (果园与林地面积之比) 呈现较强的负相关, 与 X_6 、 X_7 、 X_9 呈正相关, 这三个变量体现的是土地面积的使用效率, 因此认为第三主成分代表的是土地使用情况。

显然, 用三个主成分 F_1 、 F_2 、 F_3 代替原来 9 个变量 (X_1 , X_2 , ..., X_9) 来描述农业生态经济

系统，可以使问题更进一步简化，对解释变量的概括性也增强了。

有时，主成分分析也辅以图像帮助理解。首先，判断主成分个数时可以绘制主成分的碎石图，以成分数为横坐标，特征值即成分的方差为纵坐标，碎石图由陡峭变为平坦的转折点即为主成分选择的最佳个数。在 R 中使用函数 `screplot()` 绘制，调用格式为：

```
screplot(x, npcs = min(10, length(x$sdev)), type = c("barplot", "lines"),
        main = deparse(substitute(x)), ...)
```

`x` 是主成分分析 `princomp()` 返回的对象；`npcs` 指定画出主成分的个数；`type` 是碎石图的类型，“barplot”是条形图，“lines”是线形图；`main` 为图形添加主标题。

例如，在 R 中绘制上例的碎石图（如图 10.2 所示）。

```
>screplot(agri.pr,type="line",main="碎石图")
```

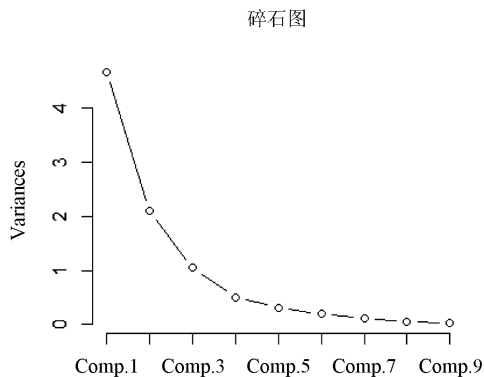


图 10.2 主成分分析的碎石图

图 10.2 中第三个主成分对应的方差由陡峭开始变得平坦，说明后几个主成分的方差贡献率很低，所以最终选择 3 个主成分即可。

另外，函数 `biplot()` 可以绘制数据关于主成分的散点图，并自动标明原坐标在主成分下的方向。其调用格式为 `biplot(x,choices=1:2)`，`x` 同样是 `princomp()` 返回的对象，参数 `choices` 选择主成分，默认为前两个。

```
>biplot(agri.pr)
```

绘制结果如图 10.3 所示。

主成分分析在市场分析方面也有很强的应用价值，例如为了研究消费者的偏好，公司常常会在市场上进行一系列调查，从而得到多种食品的消费数据。为了建立一套完整的指标体系，被调查的食品种类很多，这时就需要用主成分分析来提炼这些信息，综合原始变量。表 10.2 是我国 2003 年各地区农村居民家庭平均每人主要食品消费量，我们将利用主成分分析对食品进行分类。

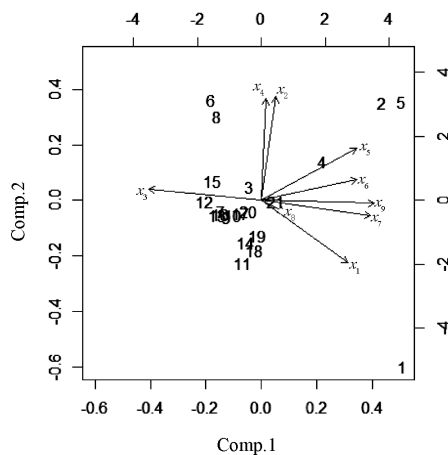


图 10.3 主成分的散点图

表 10.2 我国各地区农村居民家庭平均每人主要食品消费量

地区	粮食	蔬菜	食油	猪牛羊肉	家禽	蛋类及其制品	水产品	食糖	酒
北京	134.05	92.78	9.15	14.6	2.17	10.13	4.25	2.92	14.42
天津	150.2	69.99	10	11.07	0.84	10.8	8.35	0.72	10.14
河北	216.72	55.97	6.59	7.1	0.54	6.36	2.25	0.65	7.29
山西	218.91	80.87	5.72	5.36	0.24	6.15	0.47	1.15	2.59
内蒙	207.3	70.77	2.79	21.18	1.41	3.82	1.45	1.34	10.77
辽宁	194.39	178.59	5.9	16.45	2.51	9.59	4.49	0.73	10.8
吉林	255.99	115.2	6.27	11.42	3.23	8.64	3.6	0.75	13.64
黑龙江	195.08	111.7	7.62	7.85	2.61	6.26	3.35	0.9	15.09
上海	189.44	76.6	8.59	16.37	7.4	7.51	16.11	2.12	16.77
江苏	251.98	109.12	8.27	12.05	4.5	6.72	9.09	1.3	8.82
浙江	208.46	83.91	5.81	16.42	6.03	5.33	14.64	2.13	24.15
安徽	228.35	80.97	6.87	9.07	4.27	5.04	5.43	1.42	10.61
福建	198.27	99.92	5.19	16.51	5.14	3.55	13.3	2.35	16.84
江西	264.8	144.22	8.77	13.24	3.31	3.5	5.19	1.13	7.31
山东	229.06	118.19	6.96	8.09	2.7	11.61	4.01	1	10.81
河南	236.97	100.11	4.22	6.48	1.23	8.01	1.35	1.13	4.23
湖南	227.39	159.76	9.4	19.86	2.74	3.86	8.1	0.92	7.29
湖北	247.21	149.44	8.35	17.51	3.89	3.28	6.89	1.13	4.02
广东	233.75	130.22	6.73	22.27	10.4	2.83	13.3	2.16	3.33
广西	205.65	108.94	4.92	14.44	7.33	1.12	3.57	1.18	6.14
海南	236.31	86.61	5.7	15.4	9.77	1.31	14.75	1.24	3.88

主成分分析可以应用于多个领域，R 中也有多种方法进行主成分分析。接下来我们将上面 9 类食品命名为变量 $X_1 \sim X_9$ ，利用 R 程序包 `labdsv` 中的方法 `pca()` 进行主成分分析，将这 9 类食品综合成 4 个主成分。

第一步，将 `pca(mat, cor, dim)` 中的参数 `cor` 设为 `TRUE`，表示利用相关系数矩阵作主成分分析，即相当于使用了标准化数据，因此无需再做事前的标准化；而 `dim` 表示变量综合后的维数，即生成 3 个主成分。在 R 中输入指令：

```
> food=read.table("d:/data/food.txt",header=T) #读入数据
> food=food[,-1]
> library(labdsv)
Loading required package: mgcv
This is mgcv 1.7-22. For overview type 'help("mgcv-package")'.
Loading required package: MASS
Attaching package: 'labdsv'
The following object is masked from 'package:stats':
  density
> food.pca=pca(food,dim=4,cor=TRUE) #利用相关系数矩阵计算
```

第二步，提取分析结果。虽然函数 `pca()` 是一个简单的 `prcomp` 功能，但使用起来非常方便，我们可以根据想要的信息提取结果。使用 `summary()` 函数提取主成分分析返回的对象，直接得到衡量成分重要性的结果：4 个主成分的标准差、方差贡献以及累计方差贡献率。在本例中，4 个主成分的累计方差贡献率达到 80% 以上，可以认为它们很好地综合了原始变量的主要信息。然后使用函数 `loadings.pca()` 提取载荷系数（也可以用 `loadings()`）。

```
> summary(food.pca)
Importance of components:

                [,1]      [,2]      [,3]      [,4]
Standard deviation  1.7108133  1.4900949  1.1594286  0.89521303
Proportion of Variance  0.3252091  0.2467092  0.1493638  0.08904515
Cumulative Proportion  0.3252091  0.5719183  0.7212822  0.81032730

> loadings.pca(food.pca)
Loadings:
      PC1      PC2      PC3      PC4
X1      -0.550      -0.242
X2      -0.320      0.457
X3       0.186     -0.470
X4      0.456     -0.168      0.505
X5      0.509     -0.142     -0.263
X6     -0.329      0.408     -0.270
X7      0.501      0.119     -0.112     -0.367
X8      0.388      0.333      0.130      0.113
X9      0.141      0.493           0.201
```

第三步，使用函数 `varplot.pca()` 绘制 4 个因子的碎石图以及累计方差图（如图 10.4 所示）。

```
> op=par(mfrow=c(1,2)) #分割图形区域
> varplot.pca(food.pca)
Hit Return to Continue
```

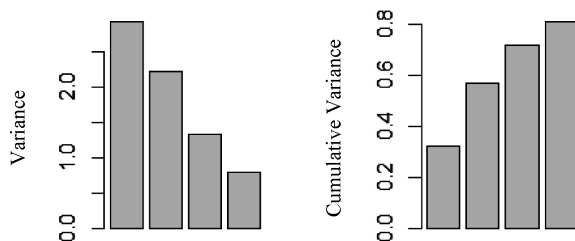


图 10.4 碎石图以及累计方差图

第四步, 根据问题的分析背景, 对各主成分赋予实际含义。这一过程通过分析主成分的载荷系数完成, 在本例中, 第一主成分对 $X_4 \sim X_8$ 的载荷系数较大, 说明第一主成分主要反映猪牛羊、家禽和水产品等方面, 可以归为肉制品类; 第二主成分对 X_1 、 X_8 、 X_9 的载荷系数较大, 分别对应粮食、食粮和酒, 归为粮食类; 第三主成分与 X_2 、 X_3 的相关性较高, 载荷系数均超过 0.6, 反映蔬菜、食油类; 第四主成分与各原始变量的相关性分配比较平均, 因此我们认为第四主成分反映的是所有食品的综合情况。这 4 个主成分对原始变量进行了概括, 根据主成分分析的结论, 下一步就可以将变量分类, 这有助于我们在市场分析中根据调查变量建立指标体系。

10.2 因子分析

与主成分分析一样, 因子分析也是一种“降维”的统计方法。它们的出发点都是变量的相关系数矩阵, 在损失较少信息的前提下, 把多个变量综合成少数几个指标来研究总体各方面信息, 并且这少数几个综合变量所代表的信息不能重叠, 即变量间不相关。

通常, 研究中得到的观察数据都是关于事物的外在特征或个别的具体特征, 这些特征的观测值存在聚合趋势, 有些变量之间存在高度的相关性, 这种高度相关性往往来源于一个共同的制约因素, 称为共同因子。如果能够在一批多维数据资料中找到 m 个因子来解释变量的大部分变异, 就是所谓的因子分析。简单地说, 就是根据相关性对变量分组, 同组内的变量之间相关性较高, 不同组间的相关性较低, 最终用少数几个因子描述指标或因素之间的联系。

10.2.1 理论模型

设 $X = (X_1, X_2, \dots, X_p)^T$ 为 p 个可观测的随机变量, 因子分析的数学模型表示为

$$X_i = a_{i1}F_1 + a_{i2}F_2 + \dots + a_{im}F_m + \varepsilon_i \quad (m \leq p)$$

写成矩阵形式

$$\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{p1} & a_{p2} & \cdots & a_{pm} \end{pmatrix} \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_m \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_p \end{pmatrix}$$

简化为 $X = AF + \varepsilon$ ，其中 F_1, F_2, \dots, F_m 为公共因子，是不可观测的变量；它们的系数称为因子载荷，即 A 是因子载荷矩阵； ε 是特殊因子，是不能被前 m 个公共因子包含的部分。通常满足以下假设

$$\begin{aligned} E(F) &= 0, \quad \text{Var}(F) = I_m \\ E(\varepsilon) &= 0, \quad \text{Var}(\varepsilon) = D = \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_p^2) \\ \text{Cov}(F, \varepsilon) &= 0 \end{aligned}$$

公共因子之间互不相关，方差为 1；特殊因子之间也互不相关，但方差不一定相等，而且满足 $\varepsilon_i \sim N(0, \sigma_i^2)$ 。

从形式上看，因子分析模型与线性回归模型相似，但它们的参数意义在本质上完全不同：回归模型中的自变量是可观测的，但因子模型中的公共因子是不可观测的潜在变量。

(1) 因子载荷 a_{ij} 的统计意义

根据数学模型 $X_i = \sum_{k=1}^m a_{ik} F_k + \varepsilon_i$ 可以推导

$$\begin{aligned} \text{Cov}(X_i, F_j) &= \text{Cov}\left(\sum_{k=1}^m a_{ik} F_k + \varepsilon_i, F_j\right) \\ &= \text{Cov}\left(\sum_{k=1}^m a_{ik} F_k, F_j\right) + \text{Cov}(\varepsilon_i, F_j) \\ &= a_{ij} \end{aligned}$$

所以，因子载荷 a_{ij} 是第 i 个随机变量与第 j 个公共因子的相关系数，其绝对值越大，相关的密切程度越高。

值得注意的是，因子载荷不是唯一的。若 Γ 是任意 m 阶正交阵，则 X 可以表示为 $X = (A\Gamma)(\Gamma^T F) + \varepsilon$ ，将 $A\Gamma$ 作为因子载荷， $\Gamma^T F$ 作为公共因子，模型仍然成立，仍满足上述假设。这种因子载荷的不唯一性，给予我们更多的选择余地，在实际使用中反而是有利的。

(2) 变量共同度的统计意义

变量 X_i 的共同度是因子载荷矩阵的第 i 行的元素的平方和, 记为 $h_i^2 = \sum_{j=1}^m a_{ij}^2$, 它反映了公共因子对 X_i 的方差贡献。

$$Var(X_i) = a_{i1}^2 Var(F_1) + \cdots + a_{im}^2 Var(F_m) + Var(\varepsilon_i), \text{ 即 } 1 = \sum_{j=1}^m a_{ij}^2 + \sigma_i^2$$

所有的公共因子和特殊因子对变量 X_i 的贡献为 1。如果 $\sum_{j=1}^m a_{ij}^2$ 非常靠近 1 且 σ_i^2 非常小, 则说明分析的效果好, 从原变量空间到公共因子空间的转化性质好。

(3) 公共因子 F_j 方差贡献的统计意义

因子载荷矩阵中各列元素的平方和 $g_j^2 = \sum_{i=1}^p a_{ij}^2$ 称为所有公共因子 F_j 对 X_i 的方差贡献和, 其是衡量 F_j 相对重要性的一个尺度, 可视为公共因子 F_j 对 X_1, X_2, \dots, X_p 的总方差贡献。

10.2.2 因子载荷矩阵的估计方法

提取因子的方法有多种, 常用的是主成分分析、主因子分析、迭代主因子分析和极大似然分析等。

其中比较特殊的方法是主成分分析法, 它是用相关矩阵 R 取代约相关矩阵 R^* , 即令所有变量的公因子方差为 1, 只要相关矩阵中无缺失数据, 此方法总是可以进行下去的; 而其他方法则与预先给定的公因子方差初值、样品含量、提取公因子的数目等因素有关, 有时会出现负特征值, 某些变量最后的共性方差大于 1 等现象。

在实际应用中, 几种方法也各有不同。主成分分析法给每个变量的公因子方差均赋予初值 1, 这等价于假定特殊因子的作用为 0, 显然其结果是比较粗略的, 但优点是计算总能顺利地进行下去。所以, 当样本量适中并且对分析结果的精度没有太过细致的要求时, 一般选用主成分法即可, 便于编程语言的实现。而主因子分析和最大似然法都能结合各变量的特点给它们的共性方差赋初值, 如果初值以及其他因素都调整得比较好, 就可以得到较好的结果, 但缺点是这种调整比较困难。实际中, 对于大样本资料, 由于最大似然法具有渐近性, 往往能够给出比主因子法更好的估计效果。

鉴于主成分分析法的特殊性, 这里我们简单介绍它的数学推导过程。

设随机向量 $X = (X_1, X_2, \dots, X_p)^T$ 的样本协方差阵为 Σ , $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$ 为 Σ 的特征根,

$U = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_p)$ 为对应的标准化特征向量，则有

$$\begin{aligned}\Sigma &= U \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_p \end{pmatrix} U^T = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_p) \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_p \end{pmatrix} \begin{pmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \vdots \\ \mathbf{u}_p^T \end{pmatrix} \\ &= \lambda_1 \mathbf{u}_1 \mathbf{u}_1^T + \lambda_2 \mathbf{u}_2 \mathbf{u}_2^T + \dots + \lambda_m \mathbf{u}_m \mathbf{u}_m^T + \lambda_{m+1} \mathbf{u}_{m+1} \mathbf{u}_{m+1}^T + \dots + \lambda_p \mathbf{u}_p \mathbf{u}_p^T \\ &= \begin{pmatrix} \sqrt{\lambda_1} \mathbf{u}_1 & \sqrt{\lambda_2} \mathbf{u}_2 & \dots & \sqrt{\lambda_p} \mathbf{u}_p \end{pmatrix} \begin{pmatrix} \sqrt{\lambda_1} \mathbf{u}_1^T \\ \sqrt{\lambda_2} \mathbf{u}_2^T \\ \vdots \\ \sqrt{\lambda_p} \mathbf{u}_p^T \end{pmatrix}\end{aligned}$$

上式给出的 Σ 表达式是精确的，但我们的目的是寻求用少数几个公共因子解释，当后面 $p-m$ 个特征根足够小时，可以略去后面的 $p-m$ 项的贡献， Σ 近似分解为

$$\hat{\Sigma} = \lambda_1 \mathbf{u}_1 \mathbf{u}_1^T + \lambda_2 \mathbf{u}_2 \mathbf{u}_2^T + \dots + \lambda_m \mathbf{u}_m \mathbf{u}_m^T + \hat{D} = \hat{A} \hat{A}^T + \hat{D},$$

在因子分析的数学模型中，特殊因子反映的是随机影响， $\varepsilon_i \sim N(0, \sigma_i^2)$ ，假定从 Σ 的分解中忽略特殊因子的方差。我们可以推导协方差矩阵

$$\begin{aligned}X &= AF + \varepsilon \\ \text{Var}(X) &= \Sigma = A \cdot \text{Var}(F) \cdot A^T + D(\varepsilon) \\ &= AA^T + \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_p^2)\end{aligned}$$

从而可以得到因子载荷矩阵 A 的估计值

$$\begin{aligned}\hat{A}_{p \times m} &= \begin{pmatrix} \sqrt{\lambda_1} \mathbf{u}_1 & \sqrt{\lambda_2} \mathbf{u}_2 & \dots & \sqrt{\lambda_m} \mathbf{u}_m \end{pmatrix} \\ \hat{D} &= \text{diag}(\hat{\sigma}_1^2, \hat{\sigma}_2^2, \dots, \hat{\sigma}_p^2) \\ \hat{\sigma}_i^2 &= s_{ii} - \sum_{j=1}^m a_{ij}^2\end{aligned}$$

与主成分分析类似的一点是，估计出因子载荷矩阵后，我们必须对得到的公共因子进行解释，结合案例背景知识说明其实际含义。但因子的参数估计方法带有随意性，有时得到的公共因子很

难与实际问题相对应，这就需要通过一个正交矩阵 Γ 作公共因子旋转，使旋转后的因子 $\Gamma^T F$ 有明确的实际含义。公共因子旋转的方法也有多个，如最大方差旋转、四次方最大旋转、等量最大法等。其中以方差最大化法最为常用，方差最大正交旋转以“使因子载荷矩阵中，各因子载荷值的总方差达到最大”作为因子载荷矩阵结构简化的准则，并且保持原公共因子的正交性和公共方差总和不变。实际应用中的最终结果是使得每个因子上的具有最大载荷的变量数最少，从而简化对因子的解释。

10.2.3 R 语言实现

R 中自带的因子分析函数 `factanal()` 采用极大似然估计方法估计因子载荷，适用于大样本量的数据分析，其调用格式为

```
factanal(x, factors, data = NULL, covmat = NULL, n.obs = NA, subset, na.action, start = NULL,
scores = c("none", "regression", "Bartlett"), rotation = "varimax", control = NULL, ...)
```

`x` 是公式或用于因子分析的数据，可以是矩阵（每行为一个样本）或数据框；`factors` 表示要生成的因子个数；`data` 指定数据集，当 `x` 为公式形式时使用；`covmat` 是样本的协方差矩阵或相关系数矩阵，使用这个参数时 `x` 可以忽略；`scores` 表示计算因子得分的方法；`rotation` 表示因子旋转方法，默认为“varimax”——方差最大旋转。

实际上，应用主成分法估计因子载荷的方法也使用得十分广泛，但 R 中仅有极大似然估计的函数 `factanal()`，因此我们可以仿照 `factanal()` 的输出结果，自己写出主成分法的因子分析函数 `factor.analysis()`。

```
factor.analysis=function(x,m){
  p=nrow(x);x.diag=diag(x);sum.rank=sum(x.diag)
  rowname=paste("X",1:p,sep="") #设置行名、列名
  colname=paste("Factor",1:m,sep="")
  #构造因子载荷矩阵A，初值设为0
  A=matrix(0,nrow=p,ncol=m,dimnames=list(rowname,colname))
  eig=eigen(x) #eig 包含两个元素，values 为特征根，vectors 为特征向量
  for(i in 1:m){
    A[,i]=sqrt(eig$values[i])*eig$vectors[,i] #填充矩阵A的值
  }
  var.A=diag(A%*%t(A)) #公共因子的方差
  rowname1=c("SS loadings","ProportionVar","Cumulative Var")
  #构造输出结果的矩阵，初值设为0
  result=matrix(0,nrow=3,ncol=m,dimnames=list(rowname1,colname))
  for(i in 1:m){
    result[1,i]=sum(A[,i]^2) #计算各因子的方差（对变量的贡献）
    result[2,i]=result[1,i]/sum.rank #计算方差贡献率
    result[3,i]=sum(result[1,1:i])/sum.rank #累计方差贡献率
  }
}
```

```

method=c("Principal Component Method")
#输出计算结果
list(method=method,loadings=A,var=cbind(common=var.A,specific=x.diag-var.A),result=result)
}

```

函数的输入参数 x 是用于因子分析的样本方差矩阵或相关系数矩阵； m 表示因子个数。函数输出一个列表，包括主成分法计算得到的因子载荷、公共因子方差和特殊因子方差，以及因子的方差、贡献率和累计方差贡献率等。

因子分析有很强的实际应用价值，在“满意度”、“幸福感”等统计调查研究中，它可以有效地分析调查问卷数据，得到综合性结论；而在企业绩效评估中，因子分析也是有效的统计方法。

我们以国内的十家股份制商业银行为研究对象，选取 11 个指标来研究它们的经营绩效，这些实际数据来源于《中国金融统计年鉴》。11 个财务指标分别为： X_1 ——流动比率； X_2 ——负债资产率； X_3 ——不良贷款率； X_4 ——贷款呆账准备率； X_5 ——资产利润率； X_6 ——资金周转率； X_7 ——投资收益率； X_8 ——营业收入费用率； X_9 ——营业收入增长率； X_{10} ——营业支出增长率； X_{11} ——利润增长率。数据存储于文本文档 bank.txt 中，部门数据如表 10.3 所示。

表 10.3 各商业银行财务指标

bank	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}
1	0.73681	-0.9572	-0.1198	0.1681	0.00009	0.00155	0.02615	4.28632	0.30796	-0.3856	-0.935
2	0.80593	-0.9598	-0.1048	0.15943	0.00409	0.00821	0.032	3.47315	0.1747	-0.1704	0.09039
3	0.77452	-0.9661	-0.0945	0.55491	0.00121	0.00357	0.0342	3.70835	0.29544	-0.2925	0.59191
...

使用我们自己编写的函数 factor.analysis()，通过主成分法计算因子载荷，将上述的 11 个指标综合成 5 个因子，来概括反映银行经营绩效的指标体系。注意这里的输入参数是数据的相关系数矩阵，所以要先用函数 cor() 计算。

```

>bank=read.table("d:/data/bank.txt",header=T)
>bank=bank[,-1]
> R=cor(bank) #计算相关系数矩阵
> options(digits=3) #结果显示3位有效数字
>factor.analysis(R,5)
$method
[1] "Principal Component Method"

$loadings
      Factor1 Factor2 Factor3 Factor4 Factor5
X1      0.394  0.2431 -0.2041  0.76100  0.3940
X2     -0.209 -0.1280  0.8928  0.12540 -0.2654
X3      0.804  0.1743  0.4772  0.20028 -0.0819

```

```

X4      0.877  0.1978  0.2827 -0.23663  0.1982
X5      0.594  0.2171 -0.0425  0.30028 -0.6327
X6      0.309  0.8681  0.2126  0.00168  0.2912
X7      0.911 -0.0434 -0.1026 -0.21235 -0.0258
X8      0.200 -0.8106  0.0252  0.52455  0.0766
X9      0.856 -0.4227  0.0227 -0.14650  0.0640
X10     -0.764  0.5250 -0.0233  0.24916 -0.2040
X11     0.644  0.2050 -0.5499 -0.00226 -0.3653

$var
common specific
X1  0.990 0.00991
X2  0.943 0.05685
X3  0.951 0.04908
X4  0.984 0.01646
X5  0.892 0.10807
X6  0.979 0.02104
X7  0.888 0.11206
X8  0.979 0.02116
X9  0.937 0.06299
X10 0.964 0.03564
X11 0.893 0.10739

$result
          Factor1 Factor2 Factor3 Factor4 Factor5
SS loadings 4.663   2.101   1.508   1.185   0.9424
Proportion Var0.424  0.191   0.137   0.108   0.0857
Cumulative Var 0.424  0.615   0.752   0.860   0.9454

```

根据 loadings 的输出结果, 我们可以写出原始变量和 5 个因子之间的线性关系式, 例如 X_1 可以表示为

$$X_1 = 0.394F_1 + 0.2431F_2 - 0.2041F_3 + 0.761F_4 + 0.394F_5$$

结合原始变量的实际含义和会计学原理来分析因子载荷矩阵, 我们可以概括银行绩效的指标体系, 将 11 个变量总结为 5 个方面。

- 因子 F_1 与 X_3 (不良贷款率)、 X_4 (贷款呆账准备率)、 X_7 (投资收益率) 和 X_9 (营业收入增长率) 的载荷系数均大于 0.8, 呈高度正相关, 因此因子 F_1 反映的是银行的成长能力;
- 因子 F_2 在 X_6 (资金周转率) 和 X_8 (营业收入费用率) 上的载荷较高, 说明 F_2 反映了银行的经营效率;
- 因子 F_3 在 X_2 (负债资产率) 和 X_{11} (利润增长率) 上的载荷很高, 说明它反映了银行的盈利水平;
- 因子 F_4 与 X_1 (流动比率) 高度相关, 因此反映的是银行的流动性和短期偿债能力; 因子 F_5 主要反映资金的使用情况。

Cumulative Var 反映了因子的累计方差贡献率，这 5 个因子的累计贡献率达到 94.54%，说明我们选取的 5 个因子已经包含了总体的大部分信息，可以较好地概括整个指标体系。

上例中包含 10 个样本、11 个观测变量，使用主成分法是最佳方案。若使用 R 自带的函数 `factanal()`，即采用极大似然估计载荷矩阵，由于数值运算的原因，相关系数矩阵的行列式值很小，R 认为它是一个不可逆的矩阵，因此计算将无法继续。但当数据量较大时，函数 `factanal()` 是不错的选择，计算结果也会更加精确。下面我们举一个市场营销的案例来说明用 `factanal()` 作因子分析的步骤。

因子分析揭示变量之间的内在关联性，简化数据维数，在市场营销领域应用广泛。营销研究结合市场问卷调查和因子分析的方法，可能涉及大量变量，其中大部分变量是相关的，因此需要将变量的数目缩减到合适的水平，以便进一步分析。最终目的是了解消费者的市场需求动态，分析产品的综合竞争力，为经营管理决策提供可靠的理论依据。

某公司想要了解消费者购买牙膏时更追求什么样的目标，于是通过商场拦访对 30 个人进行访谈，用 7 级里克特量表询问他们对以下陈述的认同程度（即 1 表示非常不同意，7 表示非常同意）。

- V_1 : 购买预防蛀牙的牙膏是重要的；
- V_2 : 我喜欢使牙齿亮泽的牙膏；
- V_3 : 牙膏应当保护牙龈；
- V_4 : 我喜欢使口气清新的牙膏；
- V_5 : 预防坏牙不是牙膏提供的一项重要功效；
- V_6 : 购买牙膏时最重要的考虑是富有魅力的牙齿；

将调查样本存储于文本文档 `yagao.txt` 中，数据示例为。

index	V1	V2	V3	V4	V5	V6
1	7	3	6	4	2	4
2	1	3	2	4	5	4
3	6	2	7	4	1	3
4	4	5	4	6	2	5
5	1	2	2	3	6	2
...						

为了在 R 中对数据作因子分析，综合原始变量信息，首先需要读入原始数据。

```
> yg=read.table("d:/data/yagao.txt",header=T)
> data=yg[,-1]
```

第二步，直接使用函数 `factanal()`。用 R 作主成分分析的步骤非常简单，只需要简短的一个命令就可以直接输出我们想要的结果。这里唯一需要确定的是 `factors` 的参数值，即提取的因子数，

本例中有 6 个变量，假设提取因子数为 2，观察因子分析的累计方差贡献率。

```
> factanal(data,factors=2)

Call:
factanal(x = data, factors = 2)

Uniquenesses:
  V1   V2   V3   V4   V5   V6
0.104 0.440 0.141 0.384 0.238 0.294

Loadings:
  Factor1 Factor2
V1  0.945
V2           0.747
V3  0.917 -0.135
V4           0.779
V5 -0.868
V6           0.838

                Factor1 Factor2
SS loadings      2.505    1.896
Proportion Var   0.417    0.316
Cumulative Var   0.417    0.733

Test of the hypothesis that 2 factors are sufficient.
The chi square statistic is 4.86 on 4 degrees of freedom.
The p-value is 0.302
```

第三步，确定因子个数。当提取因子数为 2 时，模型的累计方差贡献率为 0.733，通常我们要求累计方差贡献率达到 80% 以上，但本例中原始变量个数较少（6 个），用少数几个因子很难提取出大量的原始信息，因此 70% 以上的贡献率也可以接受。而且，因子载荷系数对应的变量关系区分明显，所以我们将因子个数确定为 2。但是若累计贡献率很低，就需要增加 `factors` 的参数值，重新作因子分析，直至得到满意的贡献率为止。

第四步，根据载荷系数矩阵，写出 2 个因子和原变量之间的线性关系式

$$F_1 = 0.945V_1 + 0.917V_3 - 0.868V_5$$

$$F_2 = 0.747V_2 - 0.135V_3 + 0.779V_4 + 0.838V_6$$

第五步，因子的实际解释意义。因子 1 和变量 V_1 、 V_3 、 V_5 有很强的相关性，载荷系数分别为 0.945、0.917 和 -0.868，分别表示“预防蛀牙”、“保护牙龈”和“预防坏牙”，因此该因子可命名为保健利益因子。因子 2 和变量 V_2 、 V_4 、 V_6 高度相关，载荷系数分别为 0.747、0.779 和 0.838，分别表示“牙齿亮泽”、“口气清新”和“富有魅力”，因此该因子可命名为社交利益因子。

第六步，得出结论，消费者想从牙膏中得到的利益有两大类，分别为保健利益和社交利益。

第 11 章

典型相关分析和对应分析

上一章我们介绍了多元统计分析的两种方法——主成分分析和因子分析，本章将继续讲述多元统计的内容，结合 R 软件在多个对象和多个指标互相关联的情况下寻找它们的统计规律。

典型相关分析研究两组变量之间的相关问题，采用主成分的思想浓缩信息，根据变量间的相关性，将两组变量的关系集中到少数几对综合变量（观测变量的线性组合）上。

对应分析探究行列变量的关系，它是一种比较特殊的多元统计分析，与我们前面介绍的几种方法有所不同。对应分析通过由定性变量构成的交互汇总表来揭示变量之间的联系，最大的特点是能把众多的样品和变量同时绘制在同一张图上。

接下来，我们学习如何用 R 软件实现这两种多元统计方法。

11.1 典型相关分析

相关关系一直是统计学的热门话题，最初研究的是两个随机变量之间的线性相关关系，用简单相关系数来衡量；之后，统计学家们发现了复相关系数，研究一个变量与多个变量之间的关系。直到 1936 年，Hotelling 提出了研究多个变量与多个变量之间相关关系的统计方法，称为典型相关分析，也是一种“降维”技术。

典型相关可以用于分析很多实际问题。例如：金融领域，研究宏观经济走势与股票市场走势之间的关系；市场分析方面，研究食品价格与销售量之间的相关性；工业方面，考察原料指标与产品质量之间的相关性，等等。

11.1.1 理论基础

典型相关分析是分析两组随机变量间线性密切程度的统计方法，是两变量间线性相关分析的

推广。其采用主成分的思想提炼信息,根据变量间的相关关系,寻找少数几对综合变量(原变量的线性组合)来替代原始观测变量,从而将两组变量的相关关系集中到少数几对综合变量的相关性分析上。

典型相关分析除要求所提取的综合变量尽可能全面地包含原始信息外,提取时还要求第一对综合变量间的相关性最大,第二对次之,依次类推。各组随机变量中既可以包含定量变量,也可以包含定性变量。

(1) 典型相关系数与典型相关变量

设 $X = (X_1, X_2, \dots, X_p)^T, Y = (Y_1, Y_2, \dots, Y_q)^T$ 是两个随机向量,利用主成分思想寻找第 i 对典型相关变量 (U_i, V_i) :

$$\begin{aligned} U_i &= a_{i1}X_1 + a_{i2}X_2 + \dots + a_{ip}X_p = a_i'X \\ V_i &= b_{i1}Y_1 + b_{i2}Y_2 + \dots + b_{iq}Y_q = b_i'Y \\ i &= 1, 2, \dots, m \quad m = \min(p, q) \end{aligned}$$

其中, a_i', b_i' 称为第 i 对典型变量系数或典型载荷。

记第一对典型相关变量间的典型相关系数为: $CanR_1 = Corr(U_1, V_1)$ (使 U_1 与 V_1 间最大相关); 第二对典型变量间的典型相关系数为: $CanR_2 = Corr(U_2, V_2)$ (与 U_1, V_1 无关; 使 U_2 与 V_2 间最大相关); 第 m 对典型变量间的典型相关系数为: $CanR_m = Corr(U_m, V_m)$ (与 $U_1, V_1, U_2, V_2, \dots, U_{m-1}, V_{m-1}$ 无关; U_m 与 V_m 间最大相关)。

(2) 典型相关变量性质

各对典型相关变量所包括的相关信息互不交叉,且满足:

① U_1, U_2, \dots, U_m 互不相关, V_1, V_2, \dots, V_m 也互不相关, 即其相关系数为

$$Corr(U_i, U_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}, \quad Corr(V_i, V_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

② 同一对典型相关变量 U_i 和 V_i 之间的相关系数为 $CanR_i$, 是除去前面 $i-1$ 个 $CanR$ 之外的最大者; 不同对的典型相关变量之间互不相关, 即

$$Corr(U_i, V_j) = \begin{cases} CanR_i & i = j \\ 0 & i \neq j \end{cases}$$

③ U_i 和 V_i 的均值为 0, 方差为 1 ($i = 1, 2, \dots, m$)。

④ $1 \geq CanR_1 \geq CanR_2 \geq \dots \geq CanR_m \geq 0$ 。

(3) 典型相关系数的求解步骤

在主成分分析中我们曾提到，为了去除量纲对方差的影响，应该在分析中对数据进行正态离差标准化。因此，我们在求解时应使用相关系数矩阵代替协方差阵，具体的计算步骤为：

① 求 X, Y 变量组的相关系数矩阵，其中， $R_{12}=R_{21}$ ，表示 X 和 Y 之间的相关系数矩阵。

$$\mathbf{R} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}$$

② 可以证明矩阵 $A = (R_{11})^{-1} R_{12} (R_{22})^{-1} R_{21}$ 和 $B = (R_{22})^{-1} R_{21} (R_{11})^{-1} R_{12}$ 有相同的非零特征根，计算 A 或 B 的特征根 λ_i 。

③ A 或 B 的特征根即为典型相关系数的平方： $\lambda_i = (\text{Can}R_i)^2, i=1, 2, \dots, m$ ，根据这一关系式，计算典型相关系数 $\text{Can}R_i$ 。

④ 求 A, B 关于 λ_i 的特征向量。设 a_i 为 A 关于 λ_i 的特征向量， b_i 为 B 关于 λ_i 的特征向量，则 a'_i, b'_i ($i=1, 2, \dots, m$) 为第 i 对典型载荷。从而可以写出第 i 对典型相关变量 (U_i, V_i) ，形式即原始变量的线性组合。

$$\begin{aligned} U_i &= a'_i X^* = a_{i1} X_1^* + a_{i2} X_2^* + \dots + a_{ip} X_p^* \\ V_i &= b'_i Y^* = b_{i1} Y_1^* + b_{i2} Y_2^* + \dots + b_{iq} Y_q^* \\ i &= 1, 2, \dots, m \quad m = \min(p, q) \end{aligned}$$

其中， X^*, Y^* 为原变量组的正态离差标准化，即 $X^* = \frac{X - \bar{X}}{S}$ 。

11.1.2 典型相关分析的应用

为了更好地应用典型相关方法做实际分析，我们首先要弄清楚几个基础问题：

① 严格地说，一个典型相关系数描述的只是一对典型变量 (U_i, V_i) 之间的相关，而不是两个变量组之间的相关，各对典型变量之间构成的多维典型相关才能共同揭示两个观测变量组之间的相关形式。

② 典型相关模型的基本假设和数据要求：

首先，要求两组变量之间为线性关系，从而每对典型变量之间也为线性关系，这是很多统计分析方法都涉及的隐性假设，因为相关系数衡量的是线性相关。

其次，每个典型变量与本组所有观测变量的关系也是线性关系。如果不是，可先线性化。例如经济水平和收入水平与其他一些反映社会发展水平的指标之间是乘法关系而非线性关系，就可

以先取对数 (\log), 再做典型相关分析。

③ 所有观测变量均以数值形式进入模型。处理定性数据时, 可按照一定形式设为虚拟变量后, 再放入典型相关模型中进行分析。

典型相关可以用于分析很多实际问题。在金融领域, 利用它可以研究宏观经济走势与股票市场走势之间的关系; 在市场分析方面, 利用它可以研究食品价格与销售量之间的关系; 在工业方面, 利用它可以考察原料指标与产品质量之间的相关性; 在教育中, 利用它可以考察学生的高考成绩与高三阶段主科成绩之间的相关性等。

11.1.3 R 语言实现

R 中进行典型相关分析的函数为 `cancor()`, 其调用格式为

```
cancor(x, y, xcenter = TRUE, ycenter = TRUE)
```

其中 x 和 y 分别是两组变量的数据矩阵; `xcenter` 和 `ycenter` 是逻辑值, 表示是否将数据中心化, 实际使用中一般均采用默认值 `TRUE`, 即将数据中心化后再进行计算。

下面介绍一个行业分析的实例。随着电子信息技术的发展, 以电话和快递等为代表的邮电业在近几十年来得到爆发性的发展, 越来越多的人选择网上或电话购物, 以快递的形式发送或接收货物。邮电业的发展与第三产业的发展密不可分, 必须适应我国当前的经济结构。为了找出邮电业和经济发展的深层次关系, 我们就可以使用典型相关分析来解决这一问题。

根据来自《中国统计年鉴》的 1995–2007 年我国国民经济数据, 利用 R 软件来做邮电业和国民经济之间的典型相关分析, 我们将采用如表 11.1 所示指标来衡量邮电业发展情况和经济发展情况。

表 11.1 数据指标

邮电业指标	经济发展指标
X_1 : 信函 (亿件)	Y_1 : 第一产业总产值 (万亿元)
X_2 : 快递 (万件)	Y_2 : 工业总产值 (万亿元)
X_3 : 移动电话年末用户 (万户)	Y_3 : 建筑业总产值 (万亿元)
X_4 : 固定电话年末用户 (万户)	Y_4 : 第三产业总产值 (万亿元)

具体数据汇总如表 11.2 所示。

表 11.2 邮电业与经济发展指标数据

年份	X_1	X_2	X_3	X_4	Y_1	Y_2	Y_3	Y_4
1995	79.55	5562.7	362.9	4070.6	12135.8	24950.6	3728.8	19978.5
1996	78.68	7096.6	685.3	5494.7	14015.4	29447.6	4387.4	23326.2

续表

年份	X_1	X_2	X_3	X_4	Y_1	Y_2	Y_3	Y_4
1997	68.55	6878.9	1323.3	7031.0	14441.9	32921.4	4621.6	26988.1
1998	65.51	7331.8	2386.3	8742.1	14817.6	34018.4	4985.8	30580.5
1999	60.52	9091.3	4329.6	10871.6	14770.0	35861.5	5172.1	33873.4
2000	77.71	11031.4	8453.3	14482.9	14944.7	40033.6	5522.3	38714.0
2001	86.93	12652.7	14522.2	18036.8	15781.3	43580.6	5931.7	44361.6
2002	106.01	14036.2	20600.5	21422.2	16537.0	47431.3	6465.5	49898.9
2003	103.84	17237.8	26995.3	26274.7	17381.7	54945.5	7490.8	56004.7
2004	82.81	19771.9	33482.4	31175.6	21412.7	65210.0	8694.3	64561.3
2005	73.51	22880.3	39340.6	35044.5	22420.0	77230.8	10133.8	73432.9
2006	71.31	26988.0	46105.8	36778.6	24040.0	91310.9	11851.1	84721.4
2007	69.50	120189.6	54730.6	36563.7	28095.0	107367.2	14014.1	100053.5

在 R 中进行典型相关分析的步骤是：

- ① 读入原始数据；
- ② 对数据进行标准化处理；
- ③ 使用函数 `cancor()` 做典型相关分析；
- ④ 得出分析结果，对结果进行描述整理；
- ⑤ 对结果进行实际含义解释，得出结论。

R 提供了执行数据中心化和标准化的函数 `scale()`，其调用格式为：

```
scale(x, center = TRUE, scale = TRUE)
```

`x` 是矩阵，提供数据；若 `center` 为数字或是与 `x` 等长的向量，那么中心化时用 `x` 减去 `center` 对应的数值；若 `center=TRUE` 则减去 `x` 的平均值，此即默认设置；若 `scale` 为数字或是与 `x` 等长的向量，则标准化时用 `x` 除以 `scale` 每个值，默认情况下 `scale=TRUE`，即 `x` 除以标准差进行中心化。通常选择默认的参数设置即可。

去除原始数据的第一列（年份）后，第 1~4 列是第一组变量、第 5~8 列是第二组变量。在 R 中输入指令：

```
> post=read.table("d:/data/post.txt",header=T)
> post=post[, -1] #去除第一列（通常为年份或序号）
> post=scale(post) #对数据做正态离差标准化
> ca=cancor(post[, 1:4], post[, 5:8])
> options(digits=4)
> ca
$cor
```

```
[1] 0.9984 0.9512 0.4436 0.3557
```

```
$xcoef
```

```
      [,1]      [,2]      [,3]      [,4]
X1 0.02705 -0.1800 -0.01501 -0.24812
X2 -0.05352 -0.2462 -0.67839 0.08167
X3 -0.13291 1.6587 1.86941 -1.41084
X4 -0.11931 -1.5133 -1.37607 1.38437
```

```
$ycoef
```

```
      [,1]      [,2]      [,3]      [,4]
Y1 -0.08637 -0.1173 -0.7481 2.2384
Y2 0.06487 1.2353 10.8194 4.7401
Y3 -0.05893 1.1485 -9.3603 -6.8260
Y4 -0.20940 -2.2606 -0.7167 -0.1437
```

```
$xcenter
```

```
      X1      X2      X3      X4
1.308e-16 5.658e-17 5.871e-18 -6.192e-17
```

```
$ycenter
```

```
      Y1      Y2      Y3      Y4
2.178e-16 7.686e-17 -9.714e-17 -1.183e-16
```

对分析结果进行描述:

① `$cor` 给出了典型相关系数,体现了各对典型变量的相关性由大到小的性质。`$xcoef` 和 `$ycoef` 分别是典型相关变量对应于原始数据 X 和 Y 的典型载荷系数。`$xcenter` 和 `$ycenter` 是数据 X 和 Y 的样本均值,由于我们事先做了标准化,因此这里是标准化数据的均值。

② 根据典型载荷系数,我们可以写出各对典型变量的表达式。根据 `$cor` 可知前两对的相关系数 $r_1 = 0.9984$, $r_2 = 0.9512$, 因此说明 U_1 与 V_1 、 U_2 与 V_2 之间具有高度的相关关系,即邮电业越发达,经济形式越好。因此我们可根据前两对典型相关变量来分析邮电业和经济发展的相关关系。

③ 前两对标准化的典型变量的线性组合是

$$\begin{cases} U_1 = 0.027X_1 - 0.054X_2 - 0.133X_3 - 0.119X_4 \\ V_1 = -0.086Y_1 + 0.065Y_2 - 0.059Y_3 - 0.209Y_4 \end{cases}$$

$$\begin{cases} U_2 = -0.18X_1 - 0.246X_2 + 1.659X_3 - 1.513X_4 \\ V_2 = -0.117Y_1 + 1.235Y_2 + 1.149Y_3 - 2.261Y_4 \end{cases}$$

在第一对典型相关变量(U_1, V_1)中, U_1 是邮电业各指标的线性组合,其中 X_3 (移动电话年末用户)和 X_4 (固定电话年末用户)比其他变量的载荷系数更大,说明移动电话和固定电话是邮

电业的主要指标，它在邮电业中占主导地位；而快递比函件的载荷要大，说明随着经济的发展，快递对邮电业的贡献在增加，函件的作用在减少。 V_1 是经济发展各指标的线性组合，其中 Y_4 （第三产业增加值）的载荷系数最大，说明第三产业是各产业中与邮电业相关联的主要指标，这符合我们事前的分析。而其他产业的载荷系数基本相当，说明它们与邮电业的关联性都差不多。

在第二对典型相关变量(U_2, V_2)中， U_2 是邮电业各指标的线性组合，其中仍是 X_3 （移动电话年末用户）和 X_4 （固定电话年末用户）的载荷系数较大；固定电话和移动电话用户量的符号相反，说明它们之间具有一定的相互抑制作用。 V_2 是经济发展各指标的线性组合，其中 Y_2 （工业）、 Y_3 （建筑业）和 Y_4 （第三产业）的载荷较大。另外，根据载荷系数的符号可以判断，其中工业、建筑业和移动电话是同方向发展的，第三产业和固定电话也是同方向发展的。

11.2 对应分析

在很多情况下，我们所关心的不仅仅是行或列变量本身，而是行变量和列变量的相互关系，这就是因子分析等方法无法解释的了。1970年法国统计学家 J.P.Benzenci 提出对应分析，也称关联分析、R-Q 型因子分析，其是一种多元相依变量统计分析技术。它通过分析由定性变量构成的交互汇总表，来揭示同一变量各类别之间的差异，以及不同变量各类别之间的对应关系，这是一种非常好的分析调查问卷的手段。

对应分析是一种视觉化的数据分析方法，其基本思想是将一个联列表的行和列中各元素的比例结构以点的形式在较低维的空间中表示出来，优点在于能够将几组看不出任何联系的数据，通过视觉上可以接受的定位图展现出来，使用起来直观、简单、方便，因此广泛应用于市场细分、产品定位、地质研究以及计算机工程等领域。

11.2.1 理论基础

对应分析是寻求样本（行）与指标（列）之间联系的低维图示法，其关键是利用一种数据变换方法，使含有 n 个样本观测值和 m 个变量的原始数据矩阵 X 变成另一个矩阵 Z ， Z 是一个过渡矩阵，在接下来的计算中使用。

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix}, \quad Z = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1m} \\ z_{21} & z_{22} & \cdots & z_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ z_{n1} & z_{n2} & \cdots & z_{nm} \end{bmatrix}$$

通过矩阵 Z ，将样本和变量有机地结合起来。首先，变量之间关系的协方差矩阵 $S_R = Z'Z$ 和样本之间关系的协方差矩阵 $S_Q = ZZ'$ 具有相同的非零特征根，它们相应的特征向量之间也有密切

的关系。

对方差矩阵 S_R 、 S_Q 进行因子分析，分别提取两个最重要的公因子 R_1 、 R_2 与 Q_1 、 Q_2 。我们采取的是一种特殊变换方法，由于 S_R 、 S_Q 具有相同的非零特征根，而这些特征根正是各因子所提供的方差，那么公因子 R_1 与 Q_1 、 R_2 与 Q_2 在本质上也是相同的。故可用相同的因子轴同时表示指标和样本，将 R_1 、 Q_1 和 R_2 、 Q_2 两组数据点画在同一个直角坐标系中，从而可以根据数据点的距离考察样本与指标之间的相互关系。

11.2.2 对应分析的步骤

设原始数据矩阵 $X = (x_{ij})_{nm}$ ， $i = 1, 2, \dots, n$ ， $j = 1, 2, \dots, m$ ， n 为样本数， m 为变量数。

① 计算过渡矩阵 $Z = (z_{ij})_{nm}$ ：

$$z_{ij} = \frac{x_{ij} - X_{i.}X_{.j} / X_{..}}{\sqrt{X_{i.}X_{.j}}}, \quad \sum_{i=1}^n \sum_{j=1}^m x_{ij} = X_{..}$$

其中 $X_{i.}$ 为第 i 行的求和， $X_{.j}$ 为第 j 列的总和， $X_{..}$ 为全部数据的总和。

② 对 $S_R = Z'Z$ 做因子分析。

计算协差阵 $S_R = Z'Z$ 的特征根 $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_m$ ，按其累积百分比 $\sum_{i=1}^p \lambda_i / \sum_{i=1}^m \lambda_i \geq 85\%$ 取前 p 个特征根，实际分析为达到降维的目标，通常取 $p=2$ 。并计算相应的标准化特征向量 u_1 、 u_2 ，从而得到因子载荷矩阵 F 。

$$F = \begin{bmatrix} u_{11}\sqrt{\lambda_1} & u_{12}\sqrt{\lambda_2} \\ u_{21}\sqrt{\lambda_1} & u_{22}\sqrt{\lambda_2} \\ \vdots & \vdots \\ u_{m1}\sqrt{\lambda_1} & u_{m2}\sqrt{\lambda_2} \end{bmatrix}$$

在两因子轴平面上作 m 个变量的散点图。

③ 对 $S_Q = ZZ'$ 做因子分析。

对上述 2 个特征根 λ_1 、 λ_2 计算 $S_Q = ZZ'$ 中相应的标准化特征向量 $v_1 = Zu_1$ 、 $v_2 = Zu_2$ ，从而得到因子载荷矩阵

$$\mathbf{G} = \begin{bmatrix} v_{11}\sqrt{\lambda_1} & v_{12}\sqrt{\lambda_2} \\ v_{21}\sqrt{\lambda_1} & v_{22}\sqrt{\lambda_2} \\ \vdots & \vdots \\ v_{n1}\sqrt{\lambda_1} & v_{n2}\sqrt{\lambda_2} \end{bmatrix}$$

继续在上面的两因子轴平面上作 n 个样本的散点图。

11.2.3 R 语言实现

R 中的程序包 MASS 提供了两个函数，`corresp()` 用于做简单的对应分析，`mca()` 用于计算多重对应分析，通常使用前者，其调用格式为

```
corresp(x, nf = 1, ...)
```

其中， \mathbf{x} 是数据矩阵； \mathbf{nf} 表示因子分析中计算因子的个数，通常取 2。

下面举例说明 `corresp()` 的使用和结果分析。一项研究汉字读写能力与数学的关系的调查取得了 232 个美国亚裔学生的数学成绩和汉字读写能力的数据。汉字读写能力指标有 3 个水平：“纯汉字”表示可以自如地进行纯汉字读写，“半汉字”表示读写中只有部分汉字（类似日文），而“纯英文”表示只能读写英文而不懂汉字；表示数学成绩的有 A、B、C、D 这 4 个水平。数据以列联表形式展示在表 11.3 中。

表 11.3 汉字读写能力与数学成绩

汉字读写能力	数学成绩				合计
	A	B	C	D	
纯汉字	47	31	2	1	81
半汉字	22	32	21	10	85
纯英文	10	11	25	20	66
合计	79	74	48	31	232

```
> ch=data.frame(A=c(47,22,10),B=c(31,32,11),C=c(2,21,25),D=c(1,10,20)) #直接给出矩阵的列名
```

```
> rownames(ch)=c("Pure-Chinese","Semi-Chinese","Pure-English") #给出矩阵的行名
```

```
> library(MASS)
```

```
> ch.ca=corresp(ch,nf=2)
```

```
> options(digits=4)
```

```
> ch.ca
```

```
First canonical correlation(s): 0.5521 0.1409
```

```
Row scores:
```


	[,1]	[,2]
Pure-Chinese	1.2069	0.6383
Semi-Chinese	-0.1368	-1.3079
Pure-English	-1.3051	0.9010

Column scores:

	[,1]	[,2]
A	0.9325	0.9196
B	0.4573	-1.1655
C	-0.2486	-0.5417
D	-1.5346	1.2773

分析结果给出了两个因子对应行变量、列变量的载荷系数。对应分析是一种可视化的多元统计方法，它主要是通过图形分析来得出结论，在 R 中我们使用函数 `biplot()` 可以提取因子分析的散点图，以直观地展示样本和变量各个水平之间的关系。

```
> biplot(ch.ca)
```

绘制结果如图 11.1 所示。

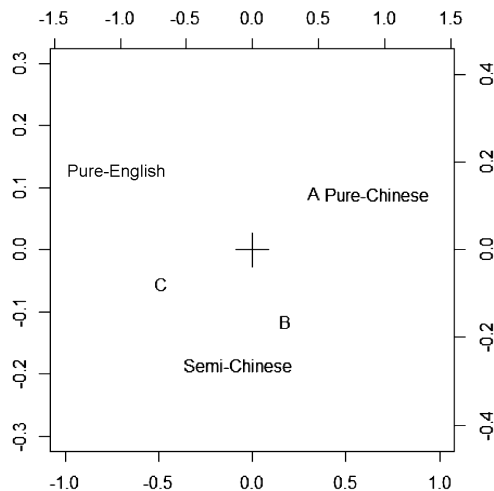


图 11.1 因子分析的散点图 1

分析图 11.1 时主要看两种散点的横坐标之间的距离，纵坐标的距离对于分析意义不大。散点“纯汉字”和数学成绩 A 最接近，说明数学好的人可以自如地进行纯汉字读写；散点“纯英文”与数学成绩 D 非常接近，说明数学差的人不会汉字只会英文；而“半汉字”介于数学成绩 B 和 C 之间，说明会部分汉字的学生数学成绩一般。

MASS 程序包的函数功能还是有限的，于是一些 R 软件使用者开发了专门用来处理对应分析的程序包，如 `ca`。`ca` 包的作者是 Michael Greenacre 和 Oleg Nenadic，该包专门用于计算并可视化简单对应分析、多重及联合对应分析。

程序包 `ca` 的主要函数汇总如表 11.4 所示。

表 11.4 程序包 `ca` 的函数

功能	函数		数据集
	简单对应分析	多重对应分析和联合对应分析	
计算	<code>ca()</code>	<code>mjca()</code>	smoke author wg93
打印、汇总	<code>print()</code> 和 <code>summary()</code>	<code>print()</code> 和 <code>summary()</code>	
绘图	<code>plot()</code> 和 <code>plot3d.ca()</code>	<code>plot()</code> 和 <code>plot3d.mjca()</code>	

对应分析广泛地应用于市场研究中，常常结合问卷调查方法，在产品定位、市场细分方面是一项非常重要的统计技术。在企业营销中，经常需要明确产品定位：什么样的消费者在使用本企业生产的产品？在不同类型的消费者心目中，哪一个品牌更受欢迎？当数据量较小时，可以使用列联表来分析不同类型的消费者在选择品牌上的差异。但是列联表存在一个问题：当变量很多且每个变量又有多个类别时，数据量很大，很难直观地发现变量间的内在联系，这时对应分析就是一种有效的解决方案。

例如，某企业对 218 名受访人员进行了收入水平和品牌选择关系的调查研究，收入水平分为高、中、低 3 个等级，品牌有 A~F 6 个。得到表 11.5 所示的调查数据，对其进行对应分析。

表 11.5 收入水平和品牌选择关系的调查数据

收入水平	品牌					
	A	B	C	D	E	F
低	2	49	4	4	15	1
中	7	7	5	49	2	7
高	16	3	23	5	5	14

```
>
brand=data.frame(low=c(2,49,4,4,15,1),medium=c(7,7,5,49,2,7),high=c(16,3,23,5,5,14))
> rownames(brand)=c("A","B","C","D","E","F")
> library(ca)
> options(digits=3)
> brand.ca=ca(brand)
> brand.ca
Principal inertias (eigenvalues):
      1      2
Value   0.530966 0.343042
Percentage 60.75% 39.25%

Rows:
      A      B      C      D      E      F
Mass   0.1147 0.271 0.147 0.266 0.101 0.1009
ChiDist 0.7704 1.026 0.906 1.029 0.738 0.7939
```

```

Inertia 0.0681 0.285 0.120 0.28 2 0.055 0.0636
Dim.    -0.7267 1.399 -0.581 -0.850 0.988 -0.8296
Dim. 2  0.9553 -0.200 1.368 -1.403 0.281 0.8786

```

Columns:

```

      low      medium      high
Mass    0.3440    0.353    0.303
ChiDist 1.0058    0.861    0.934
Inertia 0.3480    0.262    0.264
Dim. 1   1.3792   -0.778   -0.659
Dim. 2  -0.0663   -1.107    1.367

```

使用函数 `ca()` 得到的分析结果包含了更多的信息：`ChiDist` 是列联表的卡方检验结果；`Inertia` 是惯量，也就是我们所说的特征根；`Dim. 1` 和 `Dim. 2` 是提取的两个因子对行、列变量的因子载荷。可通过 `names()` 查看因子分析输出的对象列表。

```

> names(brand.ca)
[1] "sv"      "nd"      "rownames" "rowmass" "rowdist" "rowinertia"
[7] "rowcoord" "rowsup"  "colnames" "colmass" "coldist" "colinertia"
[13] "colcoord" "colsup"  "call"

```

例如，如下语句可以得到两个因子对应的行的标准坐标：

```

> brand.ca$rowcoord
      [,1] [,2]
[1,] -0.727 0.955
[2,] 1.399 -0.200
[3,] -0.581 1.368
[4,] -0.850 -1.403
[5,] 0.988 0.281
[6,] -0.830 0.879

```

接下来用 `plot()` 函数绘制因子分析的散点图，通过图形分析得到最后的结论。

```
> plot(brand.ca)
```

绘制结果如图 11.2 所示。

对应分析散点图是由品牌类别和收入类别的因子坐标值组成，从中可以看出，低收入人群倾向于选择品牌 B 和 E，中收入水平倾向于选择品牌 D，而高收入水平倾向于品牌 A、C 和 F，这样企业就完成了初步的市场定位。

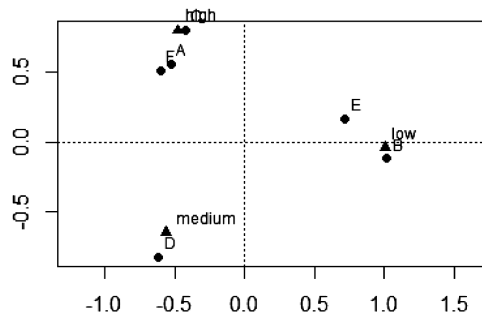


图 11.2 因子分析的散点图 2

第 12 章

判别分析和聚类分析

我们常说“物以类聚，人以群分”，分类学是人类认识世界的基础科学。20 世纪 60 年代末到 70 年代初，人们把大量精力集中于发展数值分类法，聚类分析和判别分析是其两个分支，它们和回归分析一起，并称为多元分析的三大方法。

判别分析是多元统计分析中较为成熟的一种分类方法，根据已知类别的若干样本数据，总结出客观事物分类的规律性，建立由数值指标构成的判别公式和判别准则。当遇到新的样本点时，只要根据总结出来的判别公式和判别准则，就能判别该样本点所属的类别。

聚类分析也是根据数值特征对研究对象进行分类的一种多元分析技术，其把性质相近的个体归为一类，使得同一类中的个体具有高度同质性，不同类之间的个体具有高度异质性。聚类分析简单、直观，主要应用于探索性的研究。

这两种方法都是非常实用的数据分析方法，本章将简要介绍它们的基本原理与方法，着重介绍如何应用 R 软件做数据分析。

12.1 判别分析及 R 实现

在日常生活和工作中，我们常常会遇到判别分析问题，即根据已知归类的资料确定一种判别方法，建立由数值指标构成的分类规则即判别函数，然后把这样的规则应用到未知分类的样本中，判定一个新的样品应归属于哪一类。这些已知归类的来自 k 个总体的若干个样品称为“训练样品”。

判别分析是发现事物客观规律的方法，因此具有很强的实用性。例如在气象分析中，根据已有的气象资料如温度、气压等，来判断明天是晴天还是阴天、是否下雨；又如在植物学中，新发现的一种植物，根据其各方面特征判断它应属于哪一个科目等。

判别分析的方法有多种，常用的是距离判别法（马氏距离）、Fisher 判别法和 Bayes 判别法。

12.1.1 距离判别法

距离判别法的基本思想是根据样品 x 和总体 G 的距离来判断样品所属的总体。首先要解决的问题是如何构造一个恰当的距离函数，表示样本和总体的距离。

两点间的距离公式可以从不同角度进行定义，从几何空间的距离概念出发，我们很容易想到欧式距离

$$d^2(x, G) = (x - \mu)'(x - \mu)$$

其中， μ 是 G 的均值向量。

但使用欧式距离时要注意量纲问题，当变量的测量值相差悬殊时，要先进行标准化，以消除计量单位对计算结果的影响，因此引入了马氏（Mahalanobis）距离的概念。马氏距离是变量标准化后的欧式距离，不受计量单位的影响，

$$d^2(x, G) = (x - \mu)' \Sigma^{-1} (x - \mu)$$

（1）两个总体的距离判别法

设有两个总体 G_1 和 G_2 ，两个总体的距离判别问题即为，对于一个新的样品 X ，要判断它究竟来自哪个总体。判别原则是按照就近原则归类，计算新样品 X 到 G_2 的距离与到 G_1 的距离之差，如果其值为正，则 X 属于 G_1 ，否则 X 属于 G_2 。用马氏距离给定判别规则如下：

$$\begin{cases} X \in G_1, & \text{如果 } d^2(X, G_1) < d^2(X, G_2) \\ X \in G_2, & \text{如果 } d^2(X, G_2) < d^2(X, G_1) \\ \text{待判,} & \text{如果 } d^2(X, G_1) = d^2(X, G_2) \end{cases}$$

计算中两个总体的 μ 和 Σ 真实值均未知，需要根据训练样品计算它们的极大似然估计值。特别的，如果假定两个总体的协方差阵相等，则将它们共同的 Σ 代入马氏距离的计算中，可由马氏距离之差推导出一个线性判别函数 $W(X)$ 。距离之差等于

$$d^2(X, G_2) - d^2(X, G_1) = 2(X - \frac{\mu_1 + \mu_2}{2})' \Sigma^{-1} (\mu_1 - \mu_2)$$

令 $\bar{\mu} = \frac{\mu_1 + \mu_2}{2}$ ， $\alpha = \Sigma^{-1}(\mu_1 - \mu_2)$ 称为判别系数，从而线性判别函数简化为

$$W(X) = \alpha'(X - \bar{\mu})$$

把待判样品的值代入判别函数，根据计算结果是否大于 0 得出判别结论，所以前面的判别规则就可以写成

$$\begin{cases} X \in G_1, & \text{如果 } W(X) > 0 \\ X \in G_2, & \text{如果 } W(X) < 0 \\ \text{待判}, & \text{如果 } W(X) = 0 \end{cases}$$

用线性判别函数进行判别分析非常直观，使用也方便，在实际中的应用最广泛。

(2) 多个总体的距离判别法

多个总体的距离判别仍然遵循“就近原则”，设有 k 个总体， X 是一个待判样品，它与总体 i 的距离即为判别函数，

$$\begin{aligned} d^2(X, G_i) &= (X - \mu_i)' \Sigma^{-1} (X - \mu_i) \\ &= X \Sigma^{-1} X - 2X' \Sigma^{-1} \mu_i + \mu_i' \Sigma^{-1} \mu_i \end{aligned}$$

相应的判别规则为

$$D_i = \left\{ X : d^2(X, G_i) = \min_{1 \leq j \leq k} d^2(X, G_j) \right\}, i = 1, 2, \dots, k$$

若 X 落在区域 D_i 内，那么可以判断 $X \in G_i$ 。与两总体情况类似，计算中可以考虑 $\Sigma_1 = \Sigma_2 = \dots = \Sigma_k$ 和 Σ_i 各不相同的两种情况。

12.1.2 距离判别法的 R 实现

总体来讲，进行距离判别分析时只需要知道总体的数字特征（均值和协方差矩阵），而不涉及总体的分布函数。参数未知时可用样本的均值和协方差矩阵来估计，简单实用，在 R 语言中实现起来也比较简单。首先，我们介绍两个计算距离的函数。最常用的距离函数是 `dist()`，它按照指定方法计算数据矩阵行之间的距离，默认计算欧式距离，算完后返回一个所有距离的矩阵，其调用格式为

```
dist(x, method = "euclidean", diag = FALSE, upper = FALSE, p = 2)
```

x 表示数据矩阵；`method` 用于指定计算方法，12.2 节“聚类分析及 R 实现”中我们将详细介绍；`diag` 是逻辑值，当 `diag` 为 TRUE 时，输出距离矩阵的对角线；类似的，`upper` 为 TRUE 时，输出距离矩阵的上三角部分。

另一个函数是 `mahalanobis()`，专门用来计算马氏距离，其调用格式为

```
mahalanobis(x, center, cov, inverted = FALSE, ...)
```

其中 x 是样本数据的向量或矩阵；`center` 是分布的均值；`cov` 是分布的协方差矩阵，通常使用样本值做估计；`inverted` 是逻辑值，如果为 TRUE，则 `cov` 应该包含协方差阵的逆。

随着程序包的开发，R 中已经有了可以直接进行距离判别分析的函数。目前在程序包 WMDDB 中，函数 `wmd()` 可以实现加权马氏距离判别分析，它利用了上面的函数 `mahalanobis()` 进行计算，并返回一个结果表单和准确度的报告，在两个总体和多个总体的条件下均可直接计算。我们在本节中介绍的距离判别法是传统的 Mahalanobis 距离判别方法，其将所有参数的重要性视为相同的，有时会夸大一些参数的作用，所以函数 `wmd()` 还可以通过对参数的权重进行赋值来区分每个参数的重要性。其调用格式为

```
wmd(TrnX, TrnG, Tweight = NULL, TstX = NULL, var.equal = F)
```

TrnX 是训练样品的矩阵或数据框；TrnG 用于表示已知的训练样本的分类，注意它必须是个因子向量，通过 TrnG 指定训练样本的总体可实现多个总体的判别分析；Tweight 指定权重，是个矩阵或数据框，如果没有定义权重，那么 R 将在主成分分析的基础上计算相应贡献度的百分比作为代替，当把所有参数的权重定义为等值时，就是传统的判别分析方法；TstX 是待测数据的矩阵或数据框，默认情况下是 NULL 即没有指定，可直接对训练样本进行判别分析；var.equal 指定总体是否具有相等的协方差阵，默认为 FALSE。

下面介绍一个利用判别分析研究中小企业破产模型的案例。我们选取 4 个经济指标用于判断企业处于破产状态还是正常运行状态，其中 X_1 是总负债率（现金收益/总负债）， X_2 是收益性指标（纯收入/总财产）， X_3 是短期支付能力（流动资产/流动负债）， X_4 是生产效率性指标（流动资产/纯销售额），已知 17 个破产企业（1 类）和 21 个正常运行企业（2 类）的数据资料，部分训练样品如表 12.1 所示。

表 12.1 训练样品的部分数据

X_1 总负债率	X_2 收益性指标	X_3 短期支付能力	X_4 生产效率指标	类别
-0.45	-0.41	1.09	0.45	1
-0.56	-0.31	1.51	0.16	1
0.06	0.02	1.01	0.4	1
-0.07	-0.09	1.45	0.26	1
⋮	⋮	⋮	⋮	⋮
0.38	0.11	3.27	0.55	2
0.19	0.05	2.25	0.33	2
0.32	0.07	4.24	0.63	2
⋮	⋮	⋮	⋮	⋮

根据以上信息建立判别准则，对以下 8 个待判样本进行判别分析，数据如表 12.2 所示。

表 12.2 待判样本调查数据

X_1 总负债率	X_2 收益性指标	X_3 短期支付能力	X_4 生产效率指标
0.04	0.01	1.5	0.71
-0.06	-0.06	1.37	0.4
0.07	-0.01	1.37	0.34
-0.13	-0.14	1.42	0.44
0.15	0.06	2.23	0.56
0.16	0.05	2.31	0.2
0.29	0.06	1.84	0.38
0.54	0.11	2.33	0.48

首先读入训练样品的数据，对它们计算马氏距离，函数 `mahalanobis()` 要求指定距离计算公式中的均值和协方差矩阵，因此要使用训练样品矩阵的估计值代替。

```
> B=read.table("d:/data/bankruptcy.txt",header=T)
> mu=colMeans(B) #对矩阵B的列求均值，直接得到各指标的均值
> Sx=cov(B) #计算训练样品
> distance=mahalanobis(B,mu,Sx)
> options(digits=3) #设置显示小数点格式
> distance
[1] 4.438 10.528 1.353 1.220 2.158 2.050 4.870 1.340 2.211 3.265 0.897
[12] 1.774 0.471 8.617 1.883 2.928 2.079 3.732 0.413 1.722 0.492 4.430
[23] 5.695 2.024 1.079 0.371 0.845 0.578 30.306 4.221 0.711 9.011 7.096
[34] 4.696 0.803 1.982 2.955 12.755
```

训练样本的均值和协方差阵作为总体的估计值，计算结果 `distance` 是每个训练样本距离总体的马氏距离，判别分析就是在这个结果的基础上进行的。使用程序包 `WMDB` 中的函数 `wmd()` 直接计算，首先在不指定参数 `TstX` 的情况下，对训练样品作判别分析，可以得到 38 个样本的分类判别结果、错判的样本信息以及判别分析的准确度。

```
> library(WMDB)
> G=c(rep(1,17),rep(2,21)) #生成38个训练样品的已知类别
> G=as.factor(G) #转换成因子向量，才能代入函数wmd()计算
> wmd(B,G)
      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
blong 1 1 1 1 1 1 1 1 2 1 2 2 1 2 1 1 1 2 2 2 2 2 2 2 1 2 2 2 2
      30 31 32 33 34 35 36 37 38
blong 1 2 1 2 2 2 2 2 2
[1] "num of wrong judgement"
[1] 9 11 12 14 25 30 32
[1] "samples divided to"
[1] 2 2 2 2 1 1 1
```



```
[1] "samples actually belongs to"
[1] 1 1 1 1 2 2 2
Levels: 1 2
[1] "percent of right judgement"
[1] 0.816
```

由分析结果可知, 根据已知分类的训练样品建立的判别规则, 重新应用于训练样品后, 出现了 7 个错判样品, 拥有 81.6% 的准确度。判别分析的准确度与训练样本的数据质量有关。

将待判样品的数据输入到矩阵中, 函数 `wmd()` 根据训练样品矩阵 `B` 的数据可以给出待判样品的分类情况。

```
>newdata=data.frame(X1=c(0.04,-0.06,0.07,-0.13,0.15,0.16,0.29,0.54),
+                    X2=c(0.01,-0.06,-0.01,-0.14,0.06,0.05,0.06,0.11),
+                    X3=c(1.5,1.37,1.37,1.42,2.23,2.31,1.84,2.33),
+                    X4=c(0.71,0.4,0.34,0.44,0.56,0.2,0.38,0.48))
> wmd(B,G,TstX=newdata) #TstX 表示待判样品矩阵
      1 2 3 4 5 6 7 8
blong 1 1 1 1 2 2 1 1
```

根据马氏距离判别分析得到的结果, 8 个待判样品中, 除样品 5 和 6 对应的企业处于正常运行状态外, 其余 6 个企业均处于破产状态。

12.1.3 Fisher 判别法

Fisher 判别法的基本思想是投影 (或降维): 假设预测因子有 p 个指标, 以 p 维向量 $\mathbf{x} = (x_1, x_2, \dots, x_p)'$ 表示, Fisher 判别分析就是要用 \mathbf{x} 的线性组合作为线性判别函数 $y = a_1x_1 + a_2x_2 + \dots + a_px_p$, 来代替原始的 p 个变量 x_1, x_2, \dots, x_p , 以达到降维的目的, 并找出 y 的一个临界值对样品的分类做出判别。

以两个总体 A 、 B 为例, 总体 A 有 r_1 组数据, 判别函数对应的值记为 $y_1^1, y_2^1, \dots, y_{r_1}^1$, 总体 B 的判别函数值为 $y_1^2, y_2^2, \dots, y_{r_2}^2$, 其均值为

$$\overline{y^1} = \frac{1}{r_1} \sum_{i=1}^{r_1} y_i^1, \quad \overline{y^2} = \frac{1}{r_2} \sum_{i=1}^{r_2} y_i^2$$

接下来借助方差分析的思想, 确定判别函数系数 a_1', a_2', \dots, a_r' 时要求使总体之间区别最大, 而使每个总体内部的离差平方和最小。即

- ① 组间平方和 $(\overline{y^1} - \overline{y^2})^2$ 越大越好;

② 组内平方和 $\sum_{i=1}^{r_1} (y_i^1 - \bar{y}^1)^2 + \sum_{i=1}^{r_2} (y_i^2 - \bar{y}^2)^2$ 越小越好。

这就是 Fisher 提出的最优判别准则。若满足①和②两个条件，那么比值

$$\Delta(a) = \frac{(\bar{y}^1 - \bar{y}^2)^2}{\sum_{i=1}^{r_1} (y_i^1 - \bar{y}^1)^2 + \sum_{i=1}^{r_2} (y_i^2 - \bar{y}^2)^2}$$

应该充分大，从而判别函数的系数 a_1', a_2', \dots, a_p' 为函数 $\Delta(a)$ 的极大值点。在做出判断之前，我们要给出判别的标准，即判别函数的临界值

$$y_0 = \frac{r_1 \bar{y}^1 + r_2 \bar{y}^2}{r_1 + r_2}$$

最后做判别：若一个判别对象的数据为 $(x_{01}, x_{02}, \dots, x_{0p})$ ，则可以写出它的判别函数值

$$y = c_1 x_{01} + c_2 x_{02} + \dots + c_p x_{0p}$$

判别规则为：

- ① $\bar{y}^1 > y_0$ 时，若 $y > y_0$ ，则判断该对象属于总体 A ，否则属于总体 B ；
- ② $\bar{y}^2 > y_0$ 时，若 $y > y_0$ ，则判断该对象属于总体 B ，否则属于总体 A 。

从几何的角度看，判别函数就是 p 维向量 X 在某种方向上的投影。使得变换后的数据同类别的点“尽可能聚在一起”，不同类别的点“尽可能分离”，以此达到分类的目的。

如果有多个类别，Fisher 判别可能需要两个或者更多的判别函数才能完成分类。通常判别函数的个数远远小于向量 X 的维数，才能够达到“降维”的目的，一般来说判别函数的个数等于分类的个数减一。

12.1.4 Fisher 判别法的 R 实现

R 程序包 MASS 提供了做 Fisher 判别分析的函数 `lda()`，这也是目前 R 软件中最常用的判别函数，其调用格式为

```
lda(formula, data, ..., subset, na.action)
```

`formula` 为形如 `groups ~ x1 + x2 + ...` 的公式，也就是说左侧响应变量表示分组（因子），右侧指定指标变量（非因子）；`data` 指定数据集；`subset` 是索引向量，指定训练样本；`na.action` 指定数

据中包含缺失值时应采取的行为。

如果不使用 `formula` 作为主要参数, 则 `lda()` 也可以用如下形式:

```
lda(x, grouping, ..., subset, na.action)
```

`x` 是包含解释变量的矩阵或数据框; `grouping` 是一个因子向量, 指定每个观察值的类别。这种调用方法类似于距离判别函数 `wmd()`。

继续对中小企业破产模型的案例做 Fisher 判别分析。使用 `lda()` 的第一种调用格式, 要求数据集中包含训练样本分类的变量, 所以读入数据后先做处理, 增加一列变量 `class` 表示已知类别。

```
> B=read.table("d:/data/bankruptcy.txt",header=T)
> G=c(rep(1,17),rep(2,21)) #生成 38 个训练样品的已知类别
> G=as.factor(G) #转换成因子向量
> B$class=G #将因子向量 G 存入数据框 B 中
> attach(B)
> names(B) #显示数据框 B 中的所有对象
[1] "X1" "X2" "X3" "X4" "class"
```

准备好数据后, 做判别分析:

```
> library(MASS)
> B.lda=lda(class~X1+X2+X3+X4)
> B.lda
Call:
lda(class ~ X1 + X2 + X3 + X4)

Prior probabilities of groups:
      1      2
0.4473684 0.5526316

Group means:
      X1      X2      X3      X4
1 -0.07941176 -0.088823529 1.348824 0.4300000
2  0.22571429  0.005238095 2.672857 0.4409524

Coefficients of linear discriminants:
      LD1
X1  2.9467933
X2 -1.2905768
X3  0.7666717
X4 -0.5895412
```

解释分析结果: `Group means` 给出了类别 1、2 下各个变量的均值; `Coefficients of linear discriminants` 是各判别函数对判别分类的贡献大小, 本例是两个总体的判别分析, 所以只需要一个判别函数即可。`lda()` 返回的结果 `B.lda` 包含根据训练样本建立的判别函数和判别规则, 在 `B.lda`

的基础上我们对训练样本或待判样本进行分类的判断。首先将 `lda()` 的分析结果应用于原来的训练样本进行类别的判断, 通过 R 内置函数 `predict()` 完成, 并构建一个列联表, 与真实类别进行对比。

```
> class.pre=predict(B.lda)$class #选择预测结果中的对象 class, 是预测的样本所属类别
> table(class.pre,class)
      class
class.pre  1   2
      1 15   3
      2   2 18
```

根据列联表的结果, 一共有 5 个错判的样本, 准确度为 86.8%。如果我们想要了解判别分析的效果, 准确度并不是一个绝对的判断标准, 这时我们可以对真实的样本分类和预测分类作卡方检验, 用检验的 P 值来判断判别分析的准确程度。

```
> chisq.test(class,class.pre)

Pearson's Chi-squared test with Yates' continuity correction

data:  class and class.pre
X-squared = 17.7, df = 1, p-value = 2.522e-05
```

卡方检验的 P 值远小于显著性水平 0.05, 说明判别分析的预测结果和真实值比较一致, 判别分析是可信的。接下来输入待判样本的数据, `predict()` 可以直接对新的样本做判别分析, 通过参数 `newdata` 指明待判样本的数据矩阵。

```
> newdata=data.frame(X1=c(0.04,-0.06,0.07,-0.13,0.15,0.16,0.29,0.54),
+                    X2=c(0.01,-0.06,-0.01,-0.14,0.06,0.05,0.06,0.11),
+                    X3=c(1.5,1.37,1.37,1.42,2.23,2.31,1.84,2.33),
+                    X4=c(0.71,0.4,0.34,0.44,0.56,0.2,0.38,0.48))
> predict(B.lda,newdata=newdata)
$class
[1] 1 1 1 1 2 2 2 2
Levels: 1 2

$posterior
      1      2
1 0.7449 0.255
2 0.7838 0.216
3 0.6583 0.342
4 0.8093 0.191
5 0.3658 0.634
6 0.2470 0.753
7 0.2804 0.720
8 0.0624 0.938

$x
LD1
```

```

1 -0.8120
2 -0.9333
3 -0.5794
4 -1.0216
5 0.0957
6 0.4116
7 0.3153
8 1.3042

```

可见, `predict()` 预测的结果包含 3 个对象, `$class` 是判别分析的直接结果, 给出各待判样本的分类; `$posterior` 表示各待判样本属于某一类别的概率; `$x` 是线性判别函数的具体取值。根据 Fisher 判别法得到的预测结果是, 待判样本中前 4 个样品对应的企业处于破产状态, 后 4 个企业处于正常经营状态。

12.1.5 贝叶斯判别法

贝叶斯 (Bayes) 判别法的基本思想是假定对所研究的对象已有一定的认识, 用先验概率来描述这种认识, 然后抽取样本, 用样本来修正已有的认识, 得到后验概率分布。贝叶斯判别法基于贝叶斯公式, 它可以建立一定的判断准则, 使得平均错判的损失最小。

$$P(B_i | A) = \frac{P(A | B_i)P(B_i)}{\sum P(A | B_i)P(B_i)}$$

其中, $P(B_i)$ 是 B_i 的先验概率, $P(B_i) | A$ 是 B_i 的后验概率。

设有 k 个总体 $G_i (i=1, 2, \dots, k)$, G_i 具有概率密度函数 $f_i(x)$, 根据以往的统计分析, G_i 出现的先验概率是 q_i , 则有 $q_1 + q_2 + \dots + q_k = 1$ 。判别分析的一个重要指标是误判概率, 即某样本来自总体 G_i , 但被错判到总体 G_j 的条件概率, 公式表示为

$$p(j/i) = P(X \in D_j / G_i) = \int_{D_j} f_i(x) dx, \quad i \neq j$$

其中 D_i 表示 X 的区域划分, $X \in D_i$ 时说明它属于总体 G_i 。

用 $L(j/i)$ 表示相应误判所造成的损失, 那么平均误判损失定义为

$$ECM = \sum_{i=1}^k q_i \left[\sum_{j \neq i, j=1}^k L(j/i) P(j/i) \right]$$

因此, 贝叶斯判别法的准则为, 寻找区域 D_i , 极小化平均误判损失 ECM , 即

$$D_i = \left\{ \mathbf{x} \mid h_i(\mathbf{x}) = \min_{1 \leq j \leq k} h_j(\mathbf{x}) \right\}, \quad i = 1, 2, \dots, k$$

其中， $h_j(\mathbf{x}) = \sum_{i=1}^k q_i L(j/i) f_i(\mathbf{x})$ 。

总体来说，当我们抽取了一个未知总体的样品值 \mathbf{x} ，要判断它属于哪个总体，只要先计算出 k 个按先验概率加权的平均误判损失，然后比较其大小，选取平均误判损失最小的，即可判定样品属于该总体。

12.1.6 贝叶斯判别法的 R 实现

程序包 **WMDB** 中的函数 `dbayes()` 用于实现贝叶斯判别法，它与距离判别函数 `wmd()` 属于同一个包，调用方式也很相似：

```
dbayes(TrnX, TrnG, p = rep(1, length(levels(TrnG))), TstX = NULL, var.equal = FALSE)
```

`TrnX` 是训练样品的矩阵或数据框；`TrnG` 用于表示已知的训练样本分类，是一个因子向量；`p` 是指定先验概率的向量；`TstX` 是待测数据的矩阵或数据框，默认情况下是 `NULL` 表示没有指定，则直接对训练样本进行判别分析；`var.equal` 指定总体是否具有相等的协方差阵，默认为 `FALSE`。

接下来在 R 中对本节的破产模型案例做贝叶斯判别，对训练样本做判别：

```
> library(WMDB)
> dbayes(B,G)
      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
blong  1 1 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2
      28 29 30 31 32 33 34 35 36 37 38
blong  2 2 2 2 2 2 2 2 2 2 2 2
[1] "num of wrong judgement"
[1] 3 4 5 6 7 8 9 11 12 13 15 16 17
[1] "samples divided to"
[1] 2 2 2 2 2 2 2 2 2 2 2 2
[1] "samples actually belongs to"
[1] 1 1 1 1 1 1 1 1 1 1 1 1
Levels: 1 2
[1] "percent of right judgement"
[1] 0.6578947
```

对训练样本做贝叶斯判别分析，结果出现了 13 个错判样品，准确度仅有 65.8%，3 种方法中最低，因此不宜使用贝叶斯判别法预测待判样品的类别。

12.2 聚类分析及 R 实现

聚类分析，顾名思义是研究“物以类聚”的分类学方法，又称为群分析、点群分析。它的基本思想是由于我们所研究的样品或指标之间存在不同程度的相似性，因此根据一批样品的多个观测指标，可以具体找出一些能够度量样品或指标之间相似程度的统计量，以这些统计量作为划分

类型的依据，相似程度较大的样品聚合成一类。

12.2.1 理论概述

对于同一个数据集，我们既可以对指标（变量）进行分类（即对数据的列分类），也可以对观测值（样品）进行分类（即对数据的行分类）。比如可以根据学生成绩数据对学生按照理科或文科成绩进行聚类分析，事先并不需要假定有多少类，完全可以按照数据本身的规律来分类。对变量的聚类称为 R 型聚类，而对观测值聚类称为 Q 型聚类。这两种聚类在数学上是对称的，没有什么不同。

与判别分析一样，聚类分析同样需要解决这样一个问题：如何衡量样本之间的相似程度。“距离”的概念在这里同样适用。

设每个样品有 p 个指标（变量）。把 n 个样品看成 p 维空间中的 n 个点，则两个样品间相似程度就可用 p 维空间中的两点距离公式来度量。当变量的测量值相差悬殊时，要先进行标准化，以消除计量单位对计算结果的影响。对于两点之间的距离有多种定义方式，常用的计算方法有以下几种：

$$\textcircled{1} \text{ 绝对值距离 } d_{ij} = \sum_{k=1}^p |x_{ik} - x_{jk}|;$$

$$\textcircled{2} \text{ 欧氏距离 } d_{ij} = \sqrt{\sum_{k=1}^p (x_{ik} - x_{jk})^2};$$

$$\textcircled{3} \text{ 切比雪夫距离 } d_{ij} = \max_{1 \leq k \leq p} |x_{ik} - x_{jk}|;$$

$$\textcircled{4} \text{ 闵可夫斯基距离 } d_{ij} = \left[\sum_{k=1}^p (x_{ik} - x_{jk})^q \right]^{\frac{1}{q}}, (q > 0), \text{ 当 } q=1 \text{ 时相当于绝对值距离; 当 } q=2$$

时相当于欧式距离;

$$\textcircled{5} \text{ 兰氏距离 } d_{ij} = \frac{1}{p} \sum_{k=1}^p \frac{|x_{ik} - x_{jk}|}{x_{ik} + x_{jk}};$$

12.1.2 节中介绍过的函数 `dist()` 可以完成对上述距离的计算，返回结果是一个包含所有样本间距离的矩阵。

```
dist(x, method = "euclidean", diag = FALSE, upper = FALSE, p = 2)
```

数据矩阵 x 的每一行代表一个样本；参数 `method` 用来指定不同的距离计算方法，如表 12.3 所示。

表 12.3 dist()中参数 method 的取值

参数 method 取值	对应方法
euclidean	欧式距离（默认方法）
manhattan	绝对值距离
maximum	切比雪夫距离
minkowski	闵可夫斯基距离
canberra	兰氏距离
binary	定性样本的距离

用距离来衡量样本之间的相似程度后，下一步是将距离接近的点合并为一类，这也是聚类分析的核心。系统聚类法（`hierarchical cluster`）是聚类分析诸多方法中最常用的一种，也称为“分层聚类”。它的基本思想是开始时，有多少样本点就是多少类；第一步先把最近的两类（两个点）合并成一类，然后再把剩下的最近的两类合并成一类；这样下去，每次都减少一类，直到最后只有一个大类为止。显然，越是后来合并的类，距离就越远。那么，又该如何衡量类与类之间的距离呢？R 中提供了几种方法，分别为：

- 离差平方和法（`ward`）
- 最短距离法（`single`）
- 最长距离法（`complete`），这也是 R 默认的方法
- McQuitty 相似分析法（`mcquitty`）
- 中间距离法（`median`）
- 重心法（`centroid`）

括号中对应的英文单词即 R 中的聚类分析函数 `hclust()` 对应的参数 `method` 取值，其中前两种计算方法比较受推崇。

12.2.2 R 实现举例

R 的分层聚类分析函数是 `hclust()`，其调用格式为

```
hclust(d, method = "complete", members = NULL)
```

其中，`d` 是由 `dist()` 返回的距离对象；`method` 指定系统聚类的方法，默认是 `complete` 最长距离法；`members` 默认为 `NULL`，或者是与 `d` 长度相等的向量，具体使用说明请参考 R 帮助文档。

得到聚类分析返回的对象后，使用函数 `plot()` 绘制聚类分析结果的谱系图，调用格式为

```
plot(x, labels = NULL, hang = 0.1, axes = TRUE, frame.plot = FALSE, ann = TRUE,
     main = "Cluster Dendrogram", sub = NULL, xlab = NULL, ylab = "Height", ...)
```

其中，`x` 是函数 `hclust()` 生成的对象；`labels` 取值为 `FALSE` 时，省去数据的标签，当数据量较大时做这样的处理；参数 `hang` 表明谱系图中各类别所在的位置，当 `hang` 取值为负时，将从底部开始

绘制谱系图中的类。其他参数即高级绘图函数的参数，参照第 4 章“数据的图形描述”。

函数 `plclust()` 可以代替 `plot()` 绘制聚类分析的谱系图，两者的使用方法基本相同。

聚类分析在市场细分上应用广泛，可以帮助企业对市场上的产品进行分类，从而更准确地制定营销策略。例如，某饮料企业收集了市场上 16 种饮料的热量、咖啡因、钠含量和价格 4 种变量数据，如表 12.4 所示。

表 12.4 16 种饮料数据

饮料编号	热量	咖啡因	钠	价格
1	207.2	3.3	15.5	2.8
2	36.8	5.9	12.9	3.3
3	72.2	7.3	8.2	2.4
4	36.7	0.4	10.5	4.0
5	121.7	4.1	9.2	3.5
6	89.1	4.0	10.2	3.3
7	146.7	4.3	9.7	1.8
8	57.6	2.2	13.6	2.1
9	95.9	0.0	8.5	1.3
10	199.0	0.0	10.6	3.5
11	49.8	8.0	6.3	3.7
12	16.6	4.7	6.3	1.5
13	38.5	3.7	7.7	2.0
14	0.0	4.2	13.1	2.2
15	118.8	4.7	7.2	4.1
16	107.0	0.0	8.3	4.2

读入原始数据矩阵后，首先利用函数 `dist()` 给出距离的计算结果 `d`，再利用函数 `hclust()` 对 `d` 作聚类分析，分别采用 4 种方法并绘图比较。

```
> drink=read.table("d:/data/drink.txt",header=T)
> drink=drink[,-1] #去掉第一列编号
> d=dist(drink)
> hc1=hclust(d,method="ward") #离差平方和法
> hc2=hclust(d,method="single") #最短距离法
> hc3=hclust(d,method="complete") #最长距离法
> hc4=hclust(d,method="median") #中间距离法
> opar=par(mfrow=c(2,2)) #分割绘图区域
> plot(hc1,hang=-1);plot(hc2,hang=-1);plot(hc3,hang=-1);plot(hc4,hang=-1)
> par(opar) #释放绘图区域分割
```

绘制图形如图 12.1 所示。

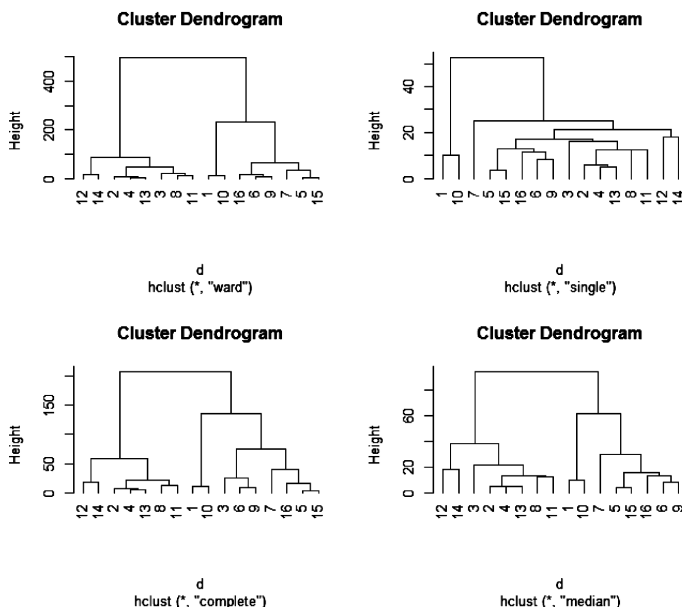


图 12.1 4 种方法下聚类分析的谱系图

4 种方法由于类与类之间距离的计算方式不同，所得到的结果也略有出入，其中最短距离法与其余三种差别最大。以离差平方和（ward）法为例，编号为 12、14、2、4、13、3、8、11 的饮料归为第一大类，其余在第二类。根据图形效果，我们把 16 种饮料细分为 4 类，通过函数 `cutree()` 可以将数据分组，以序号 1、2、3、4 分别表示 4 个类别，如：

```
> cutree(hcl,4)
[1] 1 2 2 2 3 3 3 2 3 1 2 4 2 4 3 3
```

可以看到每种饮料所属的类别，至此，通过聚类分析进行的简单市场细分就完成了。

直接使用 `plot()` 函数是绘制谱系图最简单的方式，如果想要修改谱系图样式，可以结合使用函数 `as.dendrogram()`，它的功能是将聚类分析返回的对象转换成谱系图对象，调用格式为

```
as.dendrogram(object, hang = -1, ...)
```

`object` 即函数 `hclust()` 返回的聚类分析结果对象。这时 `plot()` 的调用也有所不同：

```
plot(x, type = c("rectangle", "triangle"), center = FALSE,
     edge.root = is.leaf(x) || !is.null(attr(x,"edgetext")), nodePar = NULL, edgePar = list(),
     leaflab = c("perpendicular", "textlike", "none"), dLeaf = NULL, xlab = "", ylab = "",
     xaxt = "n", yaxt = "s", horiz = FALSE, frame.plot = FALSE, xlim, ylim, ...)
```

`x` 是由 `as.dendrogram()` 转换后的对象；`type` 表示谱系图的类型，默认值 `rectangle` 是矩阵，`triangle`

是三角形；`nodePar` 表示谱系图节点的样式，用一个 `list()` 可说明图形样式的具体信息；`horiz` 是逻辑值，当它设置为 `TRUE` 时，谱系图将水平放置。

在 R 中输入如下指令，绘制出更美观的聚类分析谱系图。

```
> drink.hc=as.dendrogram(hc1)
> par(mfrow=c(1,2))
> plot(drink.hc,type="triangle",nodePar=list(pch=c(1,NA),lab.cex=0.8))
> plot(drink.hc,nodePar=list(pch=2:1,cex=0.4*2:1,col=2:3),horiz=TRUE)
```

绘制结果如图 12.2 所示。

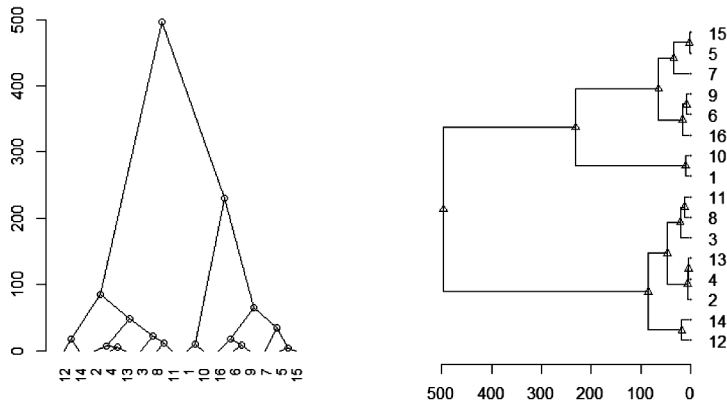


图 12.2 不同样式的谱系图

总体来讲，在 R 中实现系统聚类法的基本步骤是：

- ① 读入数据，准备好数据矩阵。
- ② 使用函数 `dist()` 计算 n 个样品两两间的距离。
- ③ 使用函数 `hclust()` 对 `dist()` 返回的结果对象做聚类分析：构造 n 个类，每个类只包含一个样品，合并距离最近的两类为一新类；计算新类与各当前类的距离，方法由参数 `method` 指定；重复上述步骤，合并距离最近的两类为新类，直到所有的类并为一类为止。
- ④ 绘制聚类谱系图；
- ⑤ 根据图形效果决定类的个数和类。

除了市场细分，聚类分析还常用来根据一些指标对全国的主要城市进行分类，从而帮助政府有针对性地对不同城市制定政策。以近年来最热门的话题——房价为例，我们可以运用聚类分析根据房价及其他一些经济指标，对城市进行分类，了解不同类别城市的特点。本例中选取了 26 座城市的 5 项经济指标，分别是：平均房价、人均 GDP、平均工资、人均可支配收入和 CPI 指数，相关数据来源于 2012 年各城市的统计公报及各主流媒体，如表 12.5 所示。

表 12.5 2012 年我国 26 座城市房价及经济指标

序号	城市	平均房价	人均 GDP	平均工资	人均可支配收入	CPI 指数
1	上海	24070	87325	5800	3349	103.6
2	北京	21080	90764	5600	3039	102.1
3	深圳	19720	125026	6200	3395	104.6
4	杭州	18600	89697	5300	2975	102
5	宁波	16860	85787	3609	3170	103.8
6	南京	16400	89967	3990	3026	102.3
7	厦门	14650	79779	4500	3131	105.7
8	广州	12440	106293	5200	3171	104.5
9	大连	11550	104628	5260	2290	102.4
10	苏州	10500	114834	4650	3127	101.7
11	天津	10260	99590	3400	2468	105.4
12	青岛	9880	83787	4380	2678	103.4
13	济南	8730	70629	3360	2714	104.5
14	无锡	8650	118705	3580	2895	105.7
15	武汉	8400	81794	2950	1978	103.8
16	海口	7860	50531	2900	1706	104
17	泉州	7650	58165	2100	2690	106
18	沈阳	7220	82654	2290	2202	101.8
19	石家庄	7180	43777	2560	1711	104.9
20	哈尔滨	7050	43128	1980	1669	103.9
21	郑州	6810	62628	3000	1873	107.5
22	西安	6760	51600	2300	2498	104.3
23	扬州	6730	65771	2300	2333	106
24	长沙	6430	90258	2900	2263	104.9
25	重庆	6300	39083	2430	1914	101.1
26	烟台	5900	75773	2500	2211	105.7

在 R 中对以上 26 个城市进行聚类分析，找出房价和经济指标具有相似性的城市，聚类方法使用“离差平方和”法。

```
> dat=read.table("d:/data/real estate.txt",header=T)
> dat=dat[, -1]
> d=dist(dat)
> hc=hclust(d,method="ward")
> plot(hc,hang=-1)
```

绘制结果如图 12.3 所示。

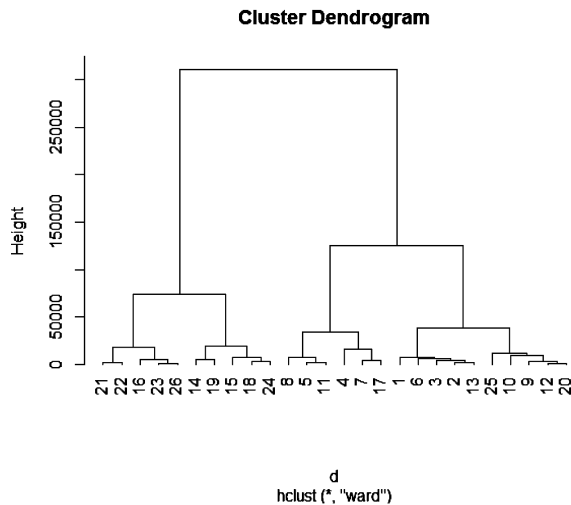


图 12.3 城市聚类分析的谱系图

为了便于在图形上显示标签，我们用序号表示对应的城市，根据谱系图的分类情况、各类别所含城市数量，我们将所有城市分为 4 类，其汇总如表 12.6 所示。

表 12.6 城市聚类分析结果

第一组	第二组	第三组	第四组	
21 郑州	14 无锡	8 广州	1 上海	25 重庆
22 西安	19 石家庄	5 宁波	6 南京	10 苏州
16 海口	15 武汉	11 天津	3 深圳	9 大连
23 扬州	18 沈阳	4 杭州	2 北京	12 青岛
26 烟台	24 长沙	7 厦门	13 济南	20 哈尔滨
		17 泉州		

根据实际的经济意义，我们可以找出每个类别中城市的相似之处，例如第一组城市的共同点的人均工资水平都不算太高，而房价也处于较低的水平；而第四组包含的城市较多，还可以分为两小类，其中第一类基本上都是一线城市，北京、上海等属于我国的经济中心城市，房价也走在全国的前列，城市居民收入较高，具有一定的购买力。类似的，对其他组别的城市也能够找出经济上的共同点，以此为基础，国家宏观调控部门可以给出不同城市应当采取的房价政策。

第 13 章

时间序列分析及 R 实现

追溯到 7000 前的古埃及有这样一个故事：古埃及人为了发现尼罗河涨落的规律，将尼罗河涨落的情况逐天记录下来，发现每年天狼星偕日升起的时候就是尼罗河泛滥的开始。尼罗河泛滥后会带来肥沃的泥土，由于掌握了这一规律，古埃及人拥有了发达的农业。这是时间序列分析最原始的应用，而现代时间序列分析主要是从经济领域的研究中发展起来的，如 Granger、Engle 等经济学家都是运用时间序列分析经济问题而获得了诺贝尔经济学奖。

时间序列分析是一种动态数据处理的统计方法，典型的假设是相邻观测值具有某种依赖性，从而基于随机过程理论和数理统计学方法，研究随机数据序列所遵从的统计规律。

本章将详细介绍时间序列分析在 R 中的实现方法，以及如何应用这一技术做预测。

13.1 时间序列的基本分析

任何事物都处于不断的发展变化中，为了研究一些现象随时间变化的数学规律，我们把某种现象发展变化的指标数据按照一定的时间顺序排列起来，就形成了时间序列，可以用 $\{X_t\}$ 表示。

13.1.1 平稳性与非平稳性

时间序列的平稳性是建模最重要的前提，有严平稳和宽平稳两种。严平稳是一种条件比较苛刻的平稳性定义，其认为只有当序列所有的统计性质都不随时间的推移而变化时，才能认为序列是平稳的。用数学模型表示，设 $\{X(t), t \in T\}$ 是一个随机过程，如果对任意的 $n \geq 1$ 和 $t_1, t_2, \dots, t_n \in T$ ，以及任意实数 τ ， n 维随机变量 $\{X(t_1), X(t_2), \dots, X(t_n)\}$ 和 $\{X(t_1 + \tau), X(t_2 + \tau), \dots, X(t_n + \tau)\}$ 有相同的联合分布函数，即

$$F_{t_1, t_2, \dots, t_n}(x_1, x_2, \dots, x_n) = F_{t_1 + \tau, t_2 + \tau, \dots, t_n + \tau}(x_1, x_2, \dots, x_n)$$

而宽平稳认为序列的统计性质由它的低阶矩决定，所以只要保证二阶矩的平稳，就能保证序列的主要性质近似平稳。设 $\{X(t), t \in T\}$ 是一个随机过程，如果它满足：

- ① $X(t)$ 是二阶矩过程；
- ② 均值函数为常数， $E[X(t)] = \mu$ ；
- ③ 相关函数 $R(t_1, t_2)$ 仅依赖 $\tau = t_1 - t_2$ ，即 $R(t_1, t_2) = E[X(t_1)X(t_2)] = B(\tau)$

则称 $X(t)$ 是宽平稳过程。通常我们所说的平稳性就是指宽平稳，而实际中常见的时间序列一般都是非平稳的。

在实际分析中，我们通常根据图形来检验平稳性，最常用的是时序图。根据平稳时间序列均值为常数的性质，平稳序列的时序图应始终在一个常数值附近随机波动，并且波动的范围有界且无明显趋势及周期特征。另一种图形是自相关图，根据平稳序列的短期相关性质，用自相关系数来反映平稳性，就是随着延迟期数的增加，平稳序列的自相关系数会迅速衰减至零。

13.1.2 R 实现的基本步骤

用 R 进行时间序列分析的最终目的是预测某种现象或某个指标的未来发展趋势，因此建立预测模型至关重要，用 R 实现时序分析的主要步骤是：

- ① 将原始数据读入到 R；
- ② 将数据存储到一个时间序列对象中；
- ③ 用这些数据绘制时间序列图；
- ④ 分解时间序列（季节性、非季节性）；
- ⑤ 做时间序列的预测。有多种预测方法，根据不同类型的数据选择适当的模型。

首先，我们介绍时间序列分析的预处理阶段，即前 3 个步骤。将数据读入 R 后，生成时间序列的函数是 `ts()`，它将会把数据存储到一个时间序列对象中，调用格式为

```
ts(data = NA, start = 1, end = numeric(), frequency = 1, deltat = 1,
   ts.eps = getOption("ts.eps"), class = , names = )
```

`data` 是时间序列观测值的一个数值向量或矩阵，数据框将通过 `data.matrix` 被强制转换成数值向量；`start` 指定收集数据的第一年及第一个间隔期，例如我们想指定第一个时间点为 1986 年二季度，则设置 `start=c(1986,2)`；`end` 指定时间序列的终止时间点；有些时间序列数据集是间隔少于一年的数据，如月度或季度数据，参数 `frequency` 就用来指定数据在一年中的频数。例如月度数据设置为 `frequency=12`，季度数据设置为 `frequency=4`；`deltat` 表示连续观测值周期的分数，如月度数据设置为 `1/12`，可以发现，`frequency` 和 `deltat` 是倒数关系，只需要设置一个即可；`names` 为多个时间序列赋名称。

```
as.ts(x, ...)
is.ts(x)
```

as.ts()将对象转换成时间序列；is.ts()用于判断对象是否为时间序列。

以我国 2002 年 1 月至 2013 年 8 月的货币供应量 M1 的数据为例，先在 R 中读入原始数据，再用函数 ts()生成时间序列，由于是月度数据，所以需要将 frequency 设为 12，同时给出起始时间 start。

```
> data=read.table("d:/data/M1.txt",header=T)
> M=ts(data$M1,frequency=12, start=c(2002,1))
> M
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2002	60577	58704	59476	60462	61248	63145	63488	64870	66800	67101	67993	70882
2003	72406	69757	71439	71322	72778	75924	76153	77033	79164	80267	80815	84119
2004	83806	83556	85816	85604	86780	88627	86780	89125	90439	90782	92387	95971
2005	97079	92815	94743	94594	95802	98601	97674	99378	100964	101752	104126	107279
2006	107251	104357	106737	106389	109219	112342	112653	114846	116814	118360	121645	126028
2007	128484	126258	127881	127678	130276	135847	136237	140993	142592	144649	148010	152519
2008	154873	150178	150867	151695	153345	154820	154992	156890	155749	157194	157827	166217
2009	165214	166150	176541	178214	182026	193138	195889	200395	201708	207546	212493	221446
2010	229589	224287	229398	233910	236498	240580	240664	244341	243822	253313	259420	266622
2011	261765	259201	266255	266767	269290	274663	270546	273394	267193	276553	281416	289848
2012	270010	270312	277998	274984	278656	287526	283091	285739	286788	293310	296883	308673
2013	311229	296103	310898	307648	310204	313500	310596	314081				

可以发现，生成的时间序列是格式整齐的数据集，使用函数 plot.ts()绘制时间序列图：

```
> plot.ts(M)
```

绘制结果如图 13.1 所示。

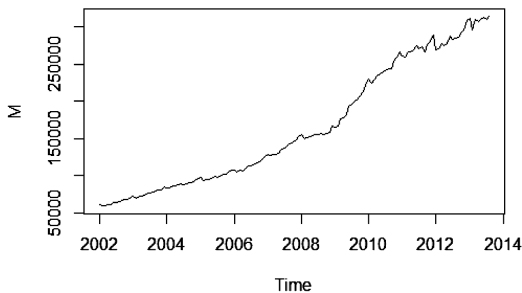


图 13.1 时间序列图

从图 13.1 中可以看出，我国的货币供应量 M1 是具有明显趋势的时间序列，并且不存在明显的季节性变动。为了对接下来各月的数值进行预测，我们需要对时间序列做进一步的分解和建模。时间序列分解是时间序列的核心内容，其可以帮助我们更好地认识和掌握现象变化发展的规律性，它将拟合出的某一构成因素的数值从时间序列中分离出去，为时间序列的预测奠定基础。

13.2 时间序列的分解

每一个现象在其变化发展过程中，每一时期都受到各种因素的影响，时间序列的形成就是这

些因素共同作用的结果，从而产生不同的变化特点，影响因素主要分为以下 4 种：

① 长期趋势 (T)。时间序列随时间而逐渐增加或减少的变化趋势，受某种长期的决定性因素影响。

② 季节变动 (S)。时间序列在一年中或固定时间内，呈现出有固定规则（周期性）的重复变动，一般是由于受到自然条件或社会条件的影响而形成。

③ 循环变动 (C)。沿着趋势线如钟摆般循环变动，循环是涨落相间的交替波动，比如经济周期。

④ 不规则变动 (e)。时间序列由于受随机因素影响所引起的变动，没有规律，由临时性、偶然性因素引起，例如受到自然灾害等不可抗力的影响。

时间序列可以表示为以上 4 个因素的函数，即 $y_t = T_t + S_t + C_t + e_t$ ，常用的有加法模型和乘法模型：

$$Y = T + S + C + e$$

$$Y = T * S * C * e$$

加法模型的基本假设是各部分对时间序列的影响相互独立，是可加的；而乘法模型假设各因素对序列的影响不是相互独立的。有些数据的量级很大，这时可以做对数变换，将乘法模型转换为加法模型，其不仅降低了数量级，也有助于提高预测的准确性。

常见的时序序列主要包括 3 个部分：整体趋势、周期部分（季节性）和噪声。接下来我们介绍如何用 R 软件来构造分解模型。

13.2.1 分解非季节性数据

非季节性的时间序列包含一个长期趋势部分和一个不规则变动部分，其时间序列的分解就是试图把一个时间序列拆分成这些部分，按照给定的分析模型，估计出趋势成分和不规则变动的具体数值。

为了估计非季节性时间序列的趋势，使用加法模型来描述，最常用的方法是移动平均法。若时间序列的各部分是乘法关系，先通过取对数 (\log) 来转换成加法关系。移动平均法是将时间序列的时距扩大，在序列中按一定项数逐项移动计算平均数，从而达到对原始序列进行修匀的目的，形成一个趋势。移动平均是测定时间序列趋势变动的基本方法，有简单移动平均法和加权移动平均法两种。

简单移动平均是对过去的 k 个历史数据求算术平均数，作为以后的趋势预测值，因此 $t+1$ 时期的预测值可以表述为

$$F_t = \frac{1}{k} \sum_{i=t-k+1}^t X_i$$

简单移动平均法对每个观测值都给予相同的权数，而加权移动平均则对近期和远期的观测值根据其重要性赋予不同的权数后再做预测。权数赋值有 3 个原则：

- ① 时间序列的波动性较大时，近期的观测值应被赋予较大的权数，远期观测值的权数依次递减；
- ② 时间序列的波动性一般时，各期的观测值的权数应相似；
- ③ 各期的权数之和必须为 1。

R 的程序包 TTR 中的函数 SMA() 和函数 WMA() 可分别实现简单移动平均和加权平均法，从而平滑时间序列数据，其调用格式为

```
SMA(x, n = 10, ...)
WMA(x, n = 10, wts = 1:n, ...)
```

其中 x 是时间序列对象，由 `ts()` 生成；参数 n 指定移动平均的跨度，跨度越大，估计得到的趋势部分越平滑； wts 是权重向量，长度必须等于 x 或 n （默认）。使用这两个函数时，一定要注意所有字母都是大写形式。

在前面的例子中，我国货币供应量的时间序列呈现非季节性，并且其随机变动的规律在整个期间内是大致不变的，因此初步判断这个序列可以被描述为一个加法模型。我们使用简单移动平均平滑来估计趋势部分，首先尝试跨度为 3 的平滑：

```
> library(TTR)
> M.SMA3=SMA(M,n=3)
> plot(M.SMA3)
```

绘制的图形如图 13.2 所示。

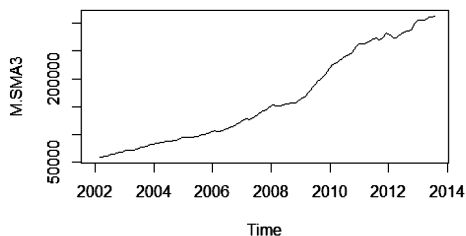


图 13.2 跨度为 3 的简单移动平均

使用跨度为 3 的简单移动平均平滑数据之后，时间序列依然包含一定的随机波动。设置的跨度越大，简单移动平均法提取的趋势部分就越平滑，而正确的跨度往往是在反复尝试中获得的。为了更加准确地估计 M 的趋势部分，我们再使用跨度为 5 的简单移动平均。

```
> M.SMA5=SMA(M,n=5)
> plot(M.SMA5)
```

绘制的图形如图 13.3 所示。

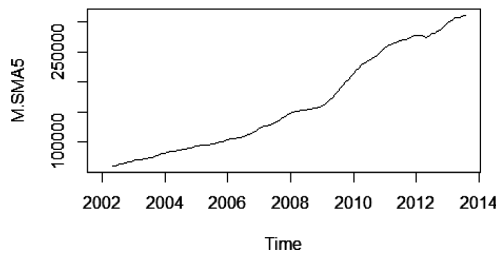


图 13.3 跨度为 5 的简单移动平均

可见，增加跨度后得到的时序图更加平滑，去除了大部分的不规则波动，趋势部分看起来更加清晰。可以发现我国的货币供应量 M1 起初在 2002 年不到 100 000，之后持续增加，直到 2013 年超过了 250 000。

13.2.2 分解季节性数据

季节性的时间序列一般包含一个趋势部分、一个季节性部分和一个不规则部分。为了解析出这 3 个部分，同样可以采用加法模型来描述，通过移动平均线完成分解。

可用 R 中的函数 `decompose()` 估计时间序列的趋势、季节性和不规则部分，其调用格式为：

```
decompose(x, type = c("additive", "multiplicative"), filter = NULL)
```

`x` 是时间序列对象；`type` 指定季节分解的形式，选择加法模型还是乘法模型；`filter` 是滤波系数。用 T 表示趋势， S 表示周期， e 表示噪声，`type=“additive”` 表示时间序列可以分解为加法模型

$$Y(t) = T(t) + S(t) + e(t)$$

`type=“multiplicative”` 表示乘法模型

$$Y(t) = T(t) * S(t) + e(t)$$

`decompose()` 返回的结果是一个列表，其中元素 `x` 是原始数据；其他元素包含估计出的季节性部分、趋势部分和不规则部分，分别对应名称 “seasonal”、“trend” 和 “random”。直接使用函数 `plot()` 对返回的结果绘图就可以分别画出时间序列中的趋势、季节性和不规则部分。

下面以某企业近年来的销售数据为例，年销售额逐年增加，但企业销售的产品具有季节性，数据如表 13.1 所示（单位：万元）。

表 13.1 某企业 2004—2009 年各季度销售额

年份	一季度	二季度	三季度	四季度
2004	362	385	432	341

续表

年份	一季度	二季度	三季度	四季度
2005	382	409	498	387
2006	473	513	582	474
2007	544	582	681	557
2008	628	707	773	592
2009	627	725	854	661

将数据输入到 R 中，并转换成时间序列，首先通过画图（如图 13.4 所示）来观察序列的特征。

```
> data=c(362,385,432,341,382,409,498,387,
+       473,513,582,474,544,582,681,557,
+       628,707,773,592,627,725,854,661)
> sales=ts(data,frequency=4,start=c(2004,1))
> plot.ts(sales)
```

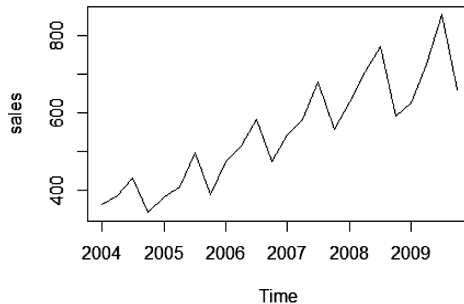


图 13.4 季节性时间序列图

从图 13.4 中可以看到数据呈现明显的季节性，销售额逐年增长也说明具有趋势性，使用函数 `decompose()` 分解趋势、季节和随机波动三部分。

```
> components=decompose(sales)
> options(digits=3) #显示小数点后3位有效数字
> components$seasonal #季节性部分
      Qtr1 Qtr2 Qtr3 Qtr4
2004 -23.2 14.6 73.4 -64.8
2005 -23.2 14.6 73.4 -64.8
2006 -23.2 14.6 73.4 -64.8
2007 -23.2 14.6 73.4 -64.8
2008 -23.2 14.6 73.4 -64.8
2009 -23.2 14.6 73.4 -64.8
```

直接对 `components` 使用函数 `plot()` 画出估计的时间序列趋势、季节性和不规则部分。

```
> plot(components)
```

绘制结果如图 13.5 所示。

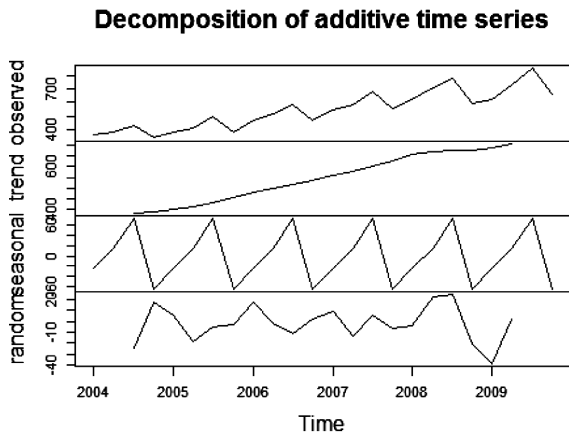


图 13.5 季节性时间序列的分解

图 13.5 中包含了 4 部分：从上向下第一个图是原始的时间序列观测值；第二个图是估计出的趋势部分；第三个图是估计出的季节性部分；最后一个是估计出的随机波动。每年的销售额在前 3 个季度逐次上升，而第四季度出现下降。

函数 `stl()` 也可以用于分解周期性时间序列，它使用 Loess 方法，计算方式更加复杂。它的调用格式为

```
stl(x, s.window, s.degree = 0,...)
```

其中 `x` 是 `ts()` 返回的时间序列对象；`s.window` 为字符串 “periodic” 或 Loess 方法提取季节的跨度，应为奇数值，要注意这一参数没有默认值；`s.degree` 是局部多项式拟合季节性提取的程度，取值为 0 或 1。`stl()` 返回的提取结果包括周期、趋势和噪声 3 个部分的分解值。

```
> components1=stl(sales,s.window="periodic")
> components1
Call:
stl(x = sales, s.window = "periodic")
```

```
Components
      seasonal trend remainder
2004 Q1    -25.4   373    14.019
2004 Q2     10.4   378     -3.160
2004 Q3     82.4   383    -33.586
2005 Q4    -67.4   389     19.711
2005 Q1    -25.4   400     6.933
2005 Q2     10.4   412    -13.145
2006 Q3     82.4   429    -13.073
2006 Q4    -67.4   455     -0.556
```

2006 Q1	-25.4	480	18.339
2006 Q2	10.4	500	2.819
2006 Q3	82.4	518	-18.370
2007 Q4	-67.4	537	4.416
2007 Q1	-25.4	559	10.647
2007 Q2	10.4	580	-8.587
2008 Q3	82.4	601	-2.274
2008 Q4	-67.4	627	-2.745
2008 Q1	-25.4	655	-1.910
2008 Q2	10.4	673	23.593
2008 Q3	82.4	677	13.985
2009 Q4	-67.4	675	-15.433
2009 Q1	-25.4	686	-33.521
2009 Q2	10.4	709	5.293
2010 Q3	82.4	731	40.386
2010 Q4	-67.4	754	-26.049

同样利用函数 `plot()` 绘图，得到的 4 个部分分别是原始时间序列图、季节性部分、趋势部分以及残差的自相关图。

```
> plot(components1)
```

绘制结果如图 13.6 所示。

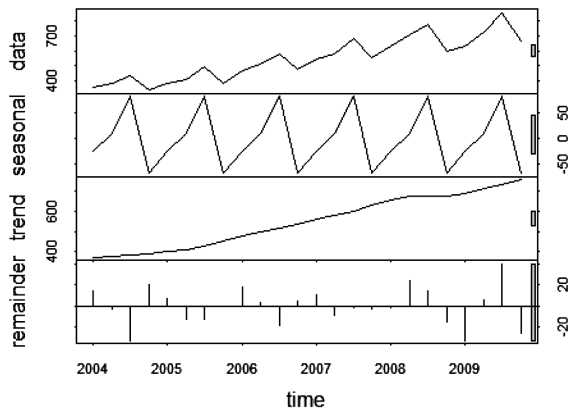


图 13.6 季节性时间序列的分解

13.3 指数平滑法预测分析

时间序列分解是对于一个时间序列数据的信息提取，接下来的预测主要使用平滑法完成，它的基本思想是移动平均。计算移动平均值时需要使用过去观察值，并且要在开始时就明确规定。每出现一个新的观察值，就要从移动平均中减去一个时间最远的观察值，再加入一个最新观察值，计算移动平均，这一新的移动平均值就作为下一期的预测值。

接下来我们介绍 3 种可以在 R 语言中实现的指数平滑方法，它们分别是简单指数平滑法（1 个参数）、Holt 双参数线性指数平滑法（2 个参数）和 Winters 线性和季节性指数平滑法（3 个参数）。看到这 3 种方法的名称，可能很多人会认为它们的模型非常复杂难懂，但事实上，这 3 个方法的原理是一样的，都基于移动平均的思想，只是参数个数设置不同，适用于不同类型的数据而已。读者不要事先被它们一长串的名字吓住，这一节我们将详细介绍它们的模型以及在 R 中实现的办法，读完之后你会发现，其实指数平滑的原理很容易理解。

在 R 中实现这 3 个方法只需一个相同的函数——`HoltWinters()`就可以完成，我们首先了解一下它的调用格式

```
HoltWinters(x, alpha = NULL, beta = NULL, gamma = NULL,
            seasonal = c("additive", "multiplicative"), start.periods = 2,
            l.start = NULL, b.start = NULL, s.start = NULL,
            optim.start = c(alpha = 0.3, beta = 0.1, gamma = 0.1), optim.control = list())
```

其中，`x` 是 `ts()` 时间序列对象；`alpha`、`beta` 和 `gamma` 是 Holt-Winters 模型的过滤系数，若 `gamma` 设置为 `FALSE`，则拟合非季节性模型，若 `beta` 也设置为 `FALSE`，则该函数将做简单指数平滑；`seasonal` 选择分解模型是加法模型还是乘法模型；`start.periods` 是初始时期，用于初始值的自动检测，必须大于或等于 2；`l.start`、`b.start` 和 `s.start` 分别表示水平、趋势和季节部分的初始值；其他详细信息参见 R 帮助文档。在下面的介绍中，我们还将详细地讲述不同方法下如何设置参数，以及 `HoltWinters()` 生成的结果。

13.3.1 简单指数平滑法

简单指数平滑法适用于对一个可用加法模型描述的，处于恒定水平并且没有季节性变动的时间序列进行短期预测，也称为一次指数平滑，“处于恒定水平”说明数据应当具有平稳性。它提供了一种方法来估计当前时点的数值，用前一期的预测值 F_t 代替 x_{t-n} ，得到预测的通式为

$$F_{t+1} = \alpha x_t + (1 - \alpha) F_t$$

由一次指数平滑法的通式可见：一次指数平滑法是一种加权预测，权数为 α 。它不需要存储全部历史数据，从而可以大大减少数据存储问题，甚至有时只需一个最新观察值、最新预测值和 α ，就可以进行预测。它提供的预测值是前一期预测值加上前期预测值中产生的误差的修正值。由参数 α 控制平滑，取值为 0~1 之间。

在 R 中使用简单指数平滑法进行短期预测，我们需要使用函数 `HoltWinters()`，将函数中的参数设置为 `beta=FALSE` 和 `gamma=FALSE`（因为 `beta` 和 `gamma` 是 Holt 指数平滑法或 Winters 指数平滑法的参数）。在简单指数平滑法中常用时间序列的第一个值作为水平的初始值，可以通过参数 `l.start` 指定其初始值。

`HoltWinters()` 默认仅给出在原始时间序列所覆盖时期内的预测，它返回一个变量列表，将生

成的预测值存储在变量 `fitted` 中，原始序列覆盖时期内的预测误差平方和存储在变量 `SSE` 中。得到预测结果后，用函数 `plot()` 直接绘制原始数据和预测值的图像，从而观察预测的效果。

`HoltWinters()` 默认仅给出原始时期的预测值，但既然是“预测”，我们更希望通过构造的模型去预测未来，这时需要用到 R 程序包 `forecast`（安装它之前还需要安装一个程序包 `fracdiff`）中的函数 `forecast()` 或 `forecast.HoltWinters()` 进行更远时间点上的预测，它可以直接调用 `HoltWinters()` 返回的预测模型对象（存储在一个变量中），调用格式为

```
forecast(object, h, level=c(80,95), fan=FALSE, ...)
```

`object` 即时间序列模型的对象；参数 `h` 指定预测时期的个数；`level` 指定预测区间的置信水平，默认给出 80% 和 95% 的预测区间；`fan` 若为 `TRUE`，将绘制扇形图。

简单指数平滑法应用的时间序列对象应该是平稳的，所以我们选取一个平稳时间序列数据在 R 中做短期预测。网页数据 <http://robjhyndman.com/tsdldata/hurst/precip1.dat> 是一个 `dat` 文件，其中包含 1813—1912 年这 100 年间伦敦每年的降雨量（单位：英尺），它是一个平稳的时间序列，使用函数 `scan()` 读入文件的数据，并绘制时序图来查看它的平稳性。在 R 中输入指令：

```
> data=scan("http://robjhyndman.com/tsdldata/hurst/precip1.dat", skip=1)
# skip=1 表示跳过第一行（基本信息）再开始读入数据
Read 100 items
> rain=ts(data, start=1813) # start 设置时间序列从 1813 年开始
> plot.ts(rain) # 画时序图
```

绘制的图形如图 13.7 所示。

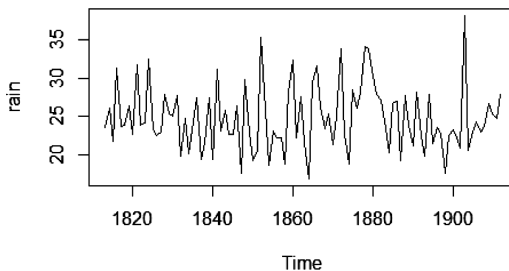


图 13.7 平稳序列的时间序列图

从数据的时间序列图中可以看出，曲线在 25 英尺左右的水平上下波动，且没有明显的趋势性和季节性，可以认为其是大致平稳的。而随机波动在整个时间序列期间也没有明显的增加或减少，可以用加法模型去描述时间序列，因此这个数据的预测应选用简单指数平滑法。模型中只包含参数 α ，所以 R 语言代码应写为：

```
> rain.pre=HoltWinters(rain, beta=FALSE, gamma=FALSE)
> rain.pre
```



```
Holt-Winters exponential smoothing without trend and without seasonal component.

Call:
HoltWinters(x = rain, beta = FALSE, gamma = FALSE)

Smoothing parameters:
  alpha: 0.02412151
  beta  : FALSE
  gamma: FALSE

Coefficients:
      [,1]
a 24.67819
```

函数 `HoltWinters()` 输出的对象 `rain.pre` 是一个变量列表，可以得到 α 的参数估计是 0.0241。没有明确设置参数 `l.start`，那么函数将默认以第一个时间点 1813 年的数值作为预测的初始水平，对 1814—1912 年降雨量所做的预测结果存储在 `rain.pre$fitted` 中。接下来用 `plot()` 绘制原始时间序列和预测值的对比图。

```
> plot(rain.pre)
```

绘制的图形如图 13.8 所示。

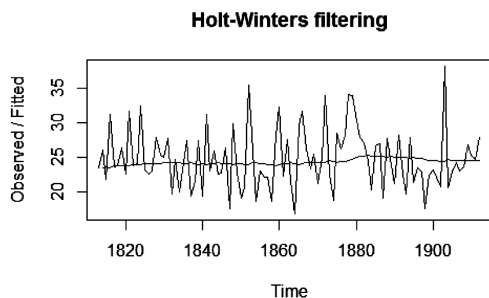


图 13.8 原始时间序列和预测值的时序图

图 13.8 中波动幅度较大的曲线是原始时间序列的图像，而中间较为平缓的曲线是预测数值。由于预测值是不包含随机波动的，因此要比原始值的时序图平滑很多，它主要反映平稳时间序列的恒定水平。图像直观地帮助我们观察预测效果，而统计的严谨性要求我们给出一个数值作为预测效果的准确计量，所以我们要计算样本内预测误差（即残差）的平方和 SSE。

```
> rain.pre$SSE
[1] 1828.855
```

这个结果说明，我们用简单指数平滑法做数据预测的误差平方和是 1828.855。

对时间序列建模的最终目的还是为了做今后时点的预测。函数 `HoltWinters()` 给出时间序列的模型后，下一步我们要利用 R 程序包 `forecast` 中的函数 `forecast.HoltWinters()` 做未来的短期预测，它直接调用上面 `HoltWinters()` 输出的对象 `rain.pre`。我们来预测未来 10 年的降雨量，设置参数 `h=10`。

```
> library(forecast)
This is forecast 4.06
> rain.pre2=forecast.HoltWinters(rain.pre, h=10)
```

预测的结果已经保存在变量 `rain.pre2` 中了, 最后用函数 `plot()` 或 `plot.forecast()` 绘制预测结果图:

```
> plot.forecast(rain.pre2,col=2)
```

绘制的图形如图 13.9 所示。

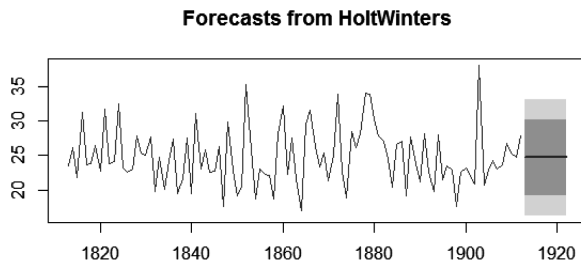


图 13.9 简单指数平滑预测图

对 `pre.rain2` 绘图可以得到两部分: 图 13.9 左侧上下波动的曲线表示原始数据的时序图; 图 13.9 右侧的阴影表示未来 10 年的预测结果, 中间的粗线条表示未来 10 年的降水量水平, 深色阴影区域为置信水平为 80% 的预测区间, 浅灰色阴影为 95% 的预测区间。

13.3.2 残差的白噪声检验

通过某种方法对时间序列建模后, 很多人不禁会问, 这个模型的效果如何? 是否能够用来预测未来时点的变化呢? 与回归分析类似, 判断时间序列模型拟合效果的重要指标也是残差。残差是样本内的预测误差, 用每个时点上的观测值减去预测值得到。在动态的时间序列模型中, 我们假设残差是服从白噪声分布的, 即

$$\varepsilon_t^{i.i.d} \sim N(0, \sigma^2)$$

简单来说, 就是残差序列是期望和方差均为常数的正态分布随机数, 是一个纯随机过程。对于一个拟合效果好的时序模型, 上述假设是成立的, 因为模型包含了序列大部分的信息, 剩下的残差仅体现随机扰动。

(1) 相关图检验

白噪声的检验方法有很多种, 首先介绍最直观的相关图检验方法。预测残差若是白噪声序列, 那么彼此独立, 序列中应该不存在显著的自相关, 所以自相关图是白噪声检验的直接方式, 我们结合上面的实例来说明如何利用图形判断残差是否存在自相关性。在伦敦降雨量的例子中, 函数 `forecast.HoltWinters()` 返回的样本残差存放于变量 `residuals` 中, 如果简单指数平滑预测模型的拟合

效果很好，那么预测残差是不相关的。

R 语言中的函数 `acf()` 可以计算残差并绘制自相关图：

```
acf(x, lag.max = NULL, type = c("correlation", "covariance", "partial"),
    plot = TRUE, na.action = na.fail, demean = TRUE, ...)
```

x 是时间序列对象；参数 `lag.max` 指定计算自相关的最大滞后阶数，默认值为 $10 * \log_{10}(N/m)$ ， N 代表观测值个数， m 代表序列个数；`type` 指定计算自相关系数的方式，默认为 “correlation”；`plot` 是逻辑值，指示是否绘制自相关图，默认为 TRUE（绘图）。

例如，绘制伦敦降雨量延迟 20 阶的残差自相关图，在 R 中输入指令：

```
> acf(rain.pre2$residuals, lag.max=20)
```

绘制结果如图 13.10 所示。

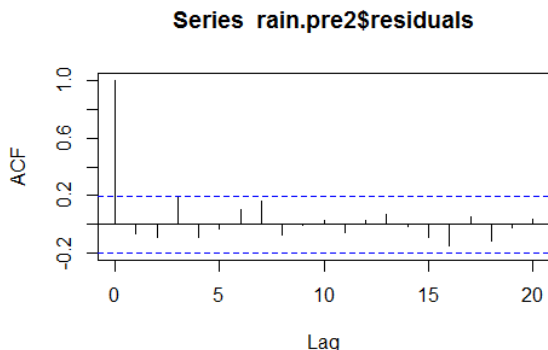


图 13.10 残差自相关图

自相关图中的两条虚线表示置信界限，是自相关系数的上下限。如果自相关系数迅速衰减，落入边界内，就可能是白噪声；若超过边界，表示存在相关关系，在哪期开始落入虚线内，就是自相关的阶数。在本例的自相关图中，所有阶的自相关系数都落入了置信界限内，我们初步判断残差序列是不存在自相关的。

（2）Ljung-Box 检验（Q 统计量）

当然，自相关图分析只能体现图形效果，而我们的判断有时会受主观影响。为了更精确地验证残差自相关在统计上是否显著，需要使用 Ljung-Box 检验。它的检验基于 Q 统计量，由 Box 和 Pierce 在 1970 年提出，所以称为 Ljung-Box 检验。检验的原假设为给定时间序列的 k 阶自相关为零，即是独立的，表示为

$$H_0: \rho_1 = \rho_2 = \cdots = \rho_k = 0 \quad H_1: \exists i \leq m, \text{使} \rho_i \neq 0$$

检验统计量由残差序列的自相关系数计算得到，即

$$Q = n(n+2) \sum_{i=1}^k \frac{\hat{\rho}_i^2}{n-i} \sim \chi^2(k)$$

在原假设成立的条件下， Q 近似服从自由度为 k 的卡方分布，它可以对小样本给出更好的卡方近似。如果 Q 统计量大于特定的临界值，那么滞后 k 阶的自相关可能显著不等于零，说明被检验序列不是独立和随机的。

这可以通过 R 中的函数 `Box.test()` 实现，调用格式为

```
Box.test(x, lag = 1, type = c("Box-Pierce", "Ljung-Box"), fitdf = 0)
```

x 是检验的时间序列；`lag` 指定计算相关系数的最大阶数；参数 `type` 指明检验类型，这里是“Ljung-Box”；`fitdf` 指明自由度。若检验结果的 P 值较大，则不能说明残差是自相关的。

```
> Box.test(rain.pre2$residuals, lag=20, type="Ljung-Box")
```

```
Box-Ljung test
```

```
data: rain.pre2$residuals
```

```
X-squared = 17.4008, df = 20, p-value = 0.6268
```

用检验结果的 P 值判断，由于 $P=0.6268$ ，故不能拒绝原假设，所以没有充分的理由说明残差序列是自相关的。经过自相关图和 Ljung-Box 检验，可以判断残差序列体现的是随机波动，因此简单指数平滑法对伦敦降雨量数据的拟合效果很好。

(3) 其他方法

用 R 检验残差的方法有很多种，这里我们再简单介绍几种常用的检验。

① Durbin-Watson 检验。简称 DW 检验，它是由 J.Durbin 和 G.S.Watson 在 1951 年提出的一种适用于小样本的检验方法，只能用于检验随机误差项具有一阶自回归形式的序列相关问题。R 中实现的函数是程序包 `car` 中的 `durbin.watson()`。

② 游程检验。检验的原假设为游程是随机的，而备择假设为游程是递增的（或递减），检验基于游程的频率。若 P 值较大，则没有足够的理由拒绝原假设，说明游程是随机的。R 中实现的函数是程序包 `tseries` 中的函数 `runs.test()`。

我们对 13.3.1 节和 13.3.2 节的内容做一个简单的小结，可以总结出在 R 中进行简单指数平滑的步骤是：

- ① 通过函数 `HoltWinters()` 对时间序列对象建立模型，得到对原始时间序列所覆盖时期的预测；
- ② 绘制原始值和预测值的对比图；
- ③ 根据模型对未来时点进行预测；
- ④ 作残差的白噪声检验，以判断模型的预测效果。

上述步骤同样适用于下面要介绍的其他两种指数平滑法。

13.3.3 Holt 指数平滑法

如果一个时间序列有增长或降低趋势、非季节性并可以用加法模型描述，那么应该选取 Holt 双参数线性指数平滑法（简称 Holt 指数平滑法）做短期预测。Holt 指数平滑法估计当前时间点的水平和斜率，直接对趋势进行平滑。计算公式为

$$\begin{aligned} F_{t+m} &= S_t + b_t m \\ S_t &= \alpha x_t + (1 - \alpha)(S_{t-1} + b_{t-1}) \\ b_t &= \beta(S_t - S_{t-1}) + (1 - \beta)b_{t-1} \end{aligned}$$

第二个等式是利用前一期的趋势值 b_{t-1} 直接修正 S_t ，而趋势项 b_t 用相邻两次平滑值之差来计算。Holt 指数平滑是由两个参数控制的： α 的作用体现在估计当前时间点的水平， β 用于估计当前时间点趋势部分的斜率。正如简单指数平滑法一样， α 和 β 参数都介于 0 和 1 之间，并且参数越接近 1，表示近期的观测值在预测中占据更大的权重。

在 R 中使用函数 `HoltWinters()` 做 Holt 指数平滑，设置参数 `gamma=FALSE`，因为 `gamma` 是 Winters 指数平滑法的参数。参数 `l.start` 和 `b.start` 分别设置水平和趋势斜率的初始值，通常水平的初始值就是其时间序列的第一个值，而斜率的初始值是第二个值减去第一个值。

具体的分析步骤与简单指数平滑法一致：首先，使用 `HoltWinters()` 进行 Holt 指数平滑后，对返回的对象应用函数 `plot()`，可以直接绘制出原始值与预测值的图形，便于比较；第二步，预测未来时点的值；最后，检验模型的效果，创建一个相关图并进行 Ljung-Box 检验分析残差是否为白噪声。

在 13.1.2 节中我们知道，我国货币供应量 M1 的数据就是一个具有递增趋势、无明显季节性的时间序列，因此对 M1 的短期预测可以通过 Holt 指数平滑法完成。

```
> data=read.table("d:/data/M1.txt",header=T)
> M=ts(data$M1,frequency=12, start=c(2002,1))
> M.pre=HoltWinters(M,gamma=FALSE)
> M.pre
Holt-Winters exponential smoothing with trend and without seasonal component.

Call:
HoltWinters(x = M, gamma = FALSE)

Smoothing parameters:
alpha: 0.6630848
beta : 0.125953
gamma: FALSE

Coefficients:
[,1]
a 314043.669
```

```
b 1783.969
```

可以得到 Holt 指数平滑模型中参数 α 的估计值是 0.663, β 的估计值是 0.126。绘图对比 Holt 平滑对原始数据的预测效果, 将函数 `plot()` 作用于对象 `M.pre` 上, 可分别绘制出原始时间序列和预测值序列两条曲线, 如图 13.11 所示。

```
> plot(M.pre)
```

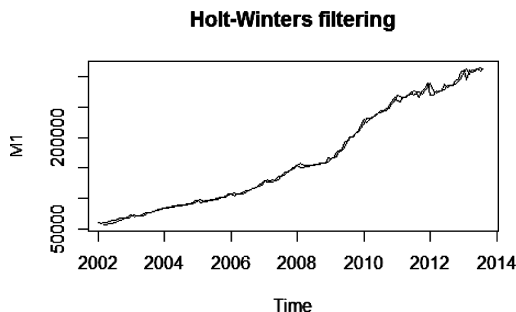


图 13.11 M1 原始值和预测值的时序图

从图形效果来看, 原始数据和预测值的差别不大, 但近几年的预测值与实际值之间波动较大, 模型中 β 的估计值是 0.126, 说明近期数据在预测中所占比重较小, 这也可以从侧面解释最近几年的预测有些出入的原因。下一步使用程序包 `forecast` 中的函数 `forecast.HoltWinters()` 预测未来 5 年 (20 个季度) M1 的值, 并用函数 `plot.forecast()` 绘制预测结果图。

```
> M.pre2=forecast.HoltWinters(M.pre,h=20) #20 个季度, 所以 h=20
> plot.forecast(M.pre2)
```

绘制结果如图 13.12 所示。

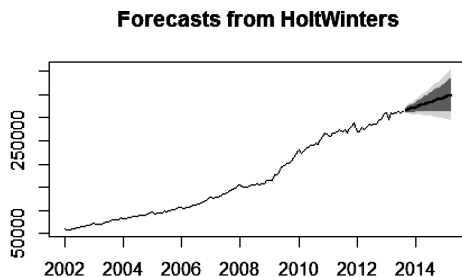


图 13.12 未来 5 年 M1 的预测图

右上角的阴影部分显示预测结果, 区域中间较粗的曲线为未来 20 个时点我国 M1 的点估计值, 深色阴影是置信水平为 80% 的预测区间, 浅色阴影为 90% 预测区间。

最后, 为检验模型的预测效果, 我们检验残差序列是否可以通过白噪声检验。

```
> acf(M.pre2$residuals,lag.max=10)
```

绘制图形如图 13.13 所示。

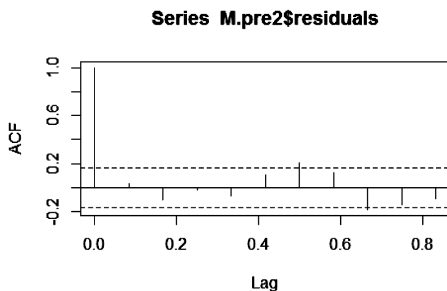


图 13.13 M1 预测模型的残差自相关图

从自相关图可以看出第 6 和第 8 阶自相关系数稍微超过了置信界限，说明残差序列有一定的自相关性，我们还需要借助 Ljung-Box 检验来做最后的判断。

```
> Box.test(M.pre2$residuals, lag=10, type="Ljung-Box")

Box-Ljung test

data: M.pre2$residuals
X-squared = 22.1802, df = 10, p-value = 0.01421
```

检验的 P 值为 0.01，在 0.05 的显著性水平下可以拒绝原假设，说明残差序列存在自相关性，Holt 指数平滑法提取的时间序列信息并不完全。

13.3.4 Winters 指数平滑法

如果一个时间序列具有增长或降低趋势，存在季节性，并且可以用加法模型描述，那么我们就可以利用 Winters 线性和季节性指数平滑法（简称 Winters 指数平滑法）对其进行短期预测。Winters 指数平滑法用于估计当前时点的水平、斜率和季节性三个部分，

$$F_{t+m} = (S_t + b_t m) I_{t-L+m}$$

其中 L 为季节的长度， I 为季节修正系数。

利用 3 个方程式，每个方程式都用于平滑模型的 3 个组成部分（平稳的、趋势的和季节性的），且都包含一个相关参数。

$$S_t = \alpha \frac{x_t}{I_{t-L}} + (1 - \alpha)(S_{t-1} + b_{t-1}), \quad 0 < \alpha < 1$$

$$b_t = \beta(S_t - S_{t-1}) + (1 - \beta)b_{t-1}, \quad 0 < \beta < 1$$

$$I_t = \gamma \frac{x_t}{S_t} + (1 - \gamma)I_{t-L}, \quad 0 < \gamma < 1$$

平滑依靠 3 个参数来控制，即 α 、 β 和 γ ，它们分别对应当前时点的水平、趋势部分的斜率和季节性部分。参数的取值都在 0~1 之间，取值越接近于 1 说明近期观测值在预测中占据的权重越大。

在 R 中实现 Winters 指数平滑法与前两种平滑方法具有相同的步骤，只是使用函数 HoltWinters() 建立模型时，参数 alpha、beta 和 gamma 均使用默认值 TRUE。

我们对 13.2.2 节中的季节性时间序列数据（企业销售额）做 Winters 指数平滑，在 R 中输入指令：

```
> data=c(362,385,432,341,382,409,498,387,
+       473,513,582,474,544,582,681,557,
+       628,707,773,592,627,725,854,661)
> sales=ts(data,frequency=4,start=c(2004,1))
> sales.pre=HoltWinters(sales)
> sales.pre
Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:
HoltWinters(x = sales)

Smoothing parameters:
alpha: 0.3612309
beta : 0.1084233
gamma: 1

Coefficients:
      [,1]
a  727.32662
b   16.84840
s1 -30.83908
s2  55.66894
s3 147.48133
s4 -66.32662
> sales.pre$SSE
[1] 19011.92
```

模型中参数值分别为 $\alpha=0.361$ ，说明历史观测值在估计当前时点的水平时占据更大比重； $\beta=0.108$ 说明估计的趋势主要依赖于历史数据，说明趋势变化并不剧烈，这一点从原始序列的时序图中也可以看出；而 $\gamma=1$ 表明当前时点季节性部分的估计仅取决于最近的观测值。Winters 指数平滑模型的总体残差平方和为 19012，除了反映估计的残差，其大小很大程度上也与数据的量纲有关。

使用函数 plot() 画出原始数据覆盖期间的实际值和预测值曲线图，比较观察模型预测效果。

```
> plot(sales.pre)
```


绘制结果如图 13.14 所示。

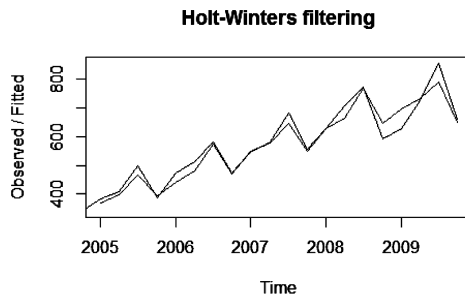


图 13.14 销售额实际值与预测值的时序图

接下来预测 2010 年的销售额数据，由于时间序列以季节为时点，因此预测时函数 `forecast.HoltWinters()` 中的参数 `h` 要设置为 4。

```
> sales.pre2=forecast.HoltWinters(sales.pre,h=4)
> plot(sales.pre2) #这里使用函数plot.forecast()的方法与plot()是一样的
```

绘制结果如图 13.15 所示。

预测图 13.15 中原始数据以曲线表示，2010 年 4 个季节销售额的预测值用散点表示，可以看出预测结果反映了序列的趋势性和季节性。最后通过自相关图和 Ljung-Box 检验来检查残差是否自相关。

```
>
> acf(sales.pre2$residuals,lag.max=8)
```

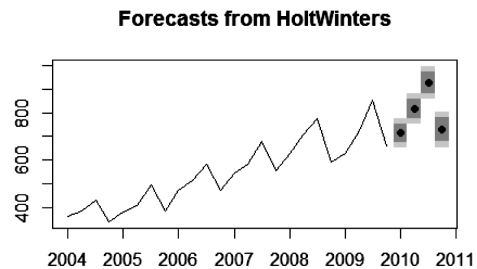


图 13.15 季节性时间序列的预测

绘制结果如图 13.16 所示。

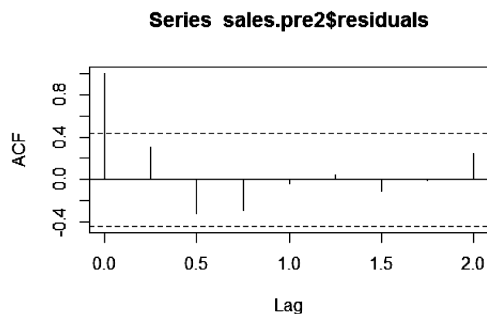


图 13.16 Winters 指数平滑的自相关图

从自相关图中可以看出，各阶残差的自相关系数都没有超过置信界限，初步判断残差不是自

相关的，再做 Ljung-Box 检验。

```
> Box.test(sales.pre2$residuals, lag=8, type="Ljung-Box")
```

```
Box-Ljung test
```

```
data: sales.pre2$residuals
X-squared = 9.4184, df = 8, p-value = 0.3082
```

检验结果得到 P 值=0.3082，没有足够理由拒绝原假设，说明残差不存在自相关性。Winters 平滑法对本例的拟合效果很好。

13.4 ARIMA 模型分析

指数平滑法要求预测残差是不相关的，而且必须服从零均值、方差不变的正态分布。这种限制很严格，但是在时间序列中，连续数值之间往往不可能没有相关性，所以在很多情况下，我们可以考虑数据之间的相关性，然后创建更好的模型进行预测，ARIMA 模型便突破了这种限制。

ARIMA 全称自回归移动平均模型，是由 Box 和 Jenkins 于 1970 年在《Time Series Analysis Forecasting and Control》一书中提出的，所以又称 Box-Jenkins 模型。该模型包含三个参数，ARIMA(p, d, q)，是一个差分自回归移动平均模型，其中 AR 表示自回归， p 是自回归的阶数；MA 表示移动平均， q 是移动平均项数，而 d 是时间序列变为平稳时所需要做的差分次数。

13.4.1 基本思想

所谓 ARIMA 模型，是指将非平稳时间序列转化为平稳时间序列，然后对平稳时间序列建立 ARMA 模型。ARIMA 模型根据原序列是否平稳以及回归中所含部分的不同，可以细分为移动平均过程 (MA)、自回归过程 (AR)、自回归移动平均过程 (ARMA) 以及 ARIMA 过程。通式写作

$$\phi(B)(1-B)^d X(t) = \theta(B)e(t)$$

$d \neq 0$ 时，原序列不是平稳的，因此方差非齐；当我们进行 d 阶差分后，差分后的序列具有方差齐性，这也是对其差分的主要原因。

R 语言中的函数 `arima.sim()` 可以产生一个模拟的 ARIMA 过程，它的调用格式为

```
arima.sim(model, n, rand.gen = rnorm, innov = rand.gen(n, ...),
           n.start = NA, start.innov = rand.gen(n.start, ...), ...)
```

`model` 是一个列表，包含对象 `ar` 和 `ma` 分别给出 AR 和 MA 的系数，`order` 设置 p 、 d 、 q 的值，一个空的列表将给出 ARIMA(0,0,0) 模型，也就是一个白噪声序列；参数 `n` 输出序列的长度，其他参数的设置请参考 R 帮助文档。

模拟时间序列并不是很常用，实际分析中我们更注重时间序列的拟合及预测。在 R 中进行时

间序列的 ARIMA 建模的步骤是：

- ① 时间序列的平稳化处理；
- ② 根据自相关和偏自相关图选择阶数，建立相应的模型；
- ③ 选好阶数后，进行参数估计；
- ④ 利用建好的模型作预测分析，并诊断残差序列是否为白噪声。

下面，我们使用 R 语言一步一步来实现上述操作。

13.4.2 平稳化处理

ARIMA 模型要在平稳时间序列的基础上建模，一般来讲，经济运行的时间序列都是非平稳序列，需要事先做平稳化处理。如果拿到的数据序列是非平稳的，并存在一定的增长或下降趋势，则应该对数据进行差分处理，直到处理后数据的自相关系数和偏相关系数均不再显著地异于零，也就说明平稳化完成了。通过这一阶段我们可以确定参数 d 的取值：如果必须对时间序列做 d 阶差分才能得到一个平稳序列，那么我们就应该选取 ARIMA(p, d, q) 模型，其中 d 是差分的阶数。

对 13.3.3 节中介绍的我国货币供应量 M1，Holt 指数平滑模型的预测效果并不好，我们尝试用 ARIMA 模型对其建模预测。首先绘制时间序列图观察序列的平稳性：

```
> data=read.table("d:/data/M1.txt",header=T)
> M=ts(data$M1,frequency=12, start=c(2002,1))
> plot.ts(M)
```

绘制结果如图 13.17 所示。

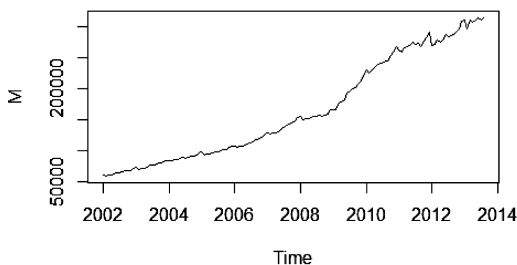


图 13.17 我国货币供应量 M1 的时间序列图

从图中可以看出 M1 的时间序列是具有明显趋势的非平稳序列，需要进行差分平稳化。R 中的函数 `diff()` 用于对时间序列作差分，调用格式为

```
diff(x, lag = 1, differences = 1, ...)
```

x 是待差分的数值向量或矩阵； lag 指定差分的阶数，默认作一阶差分。首先对 M1 作一阶差分并绘制差分序列图：

```
> M.diff=diff(M)
```

```
> plot(M.diff)
```

绘制结果如图 13.18 所示。

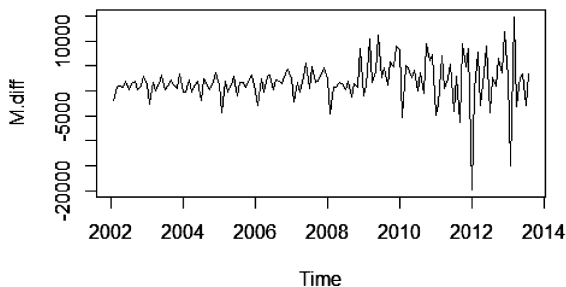


图 13.18 一阶差分序列图

从图 13.18 中看，一阶差分序列去掉了趋势，明显变得平稳了，但最近时期的波动性较大，说明存在异方差性，我们对原始数据取对数可以消除一定的异方差性，然后再作一阶差分，并用自相关图来检验差分序列是否平稳：

```
> logM.diff=diff(log(M))
> op=par(mfrow=c(1,2))
> plot(diff(logM.diff))
> acf(logM.diff,lag.max=10)
```

绘制结果如图 13.19 所示。

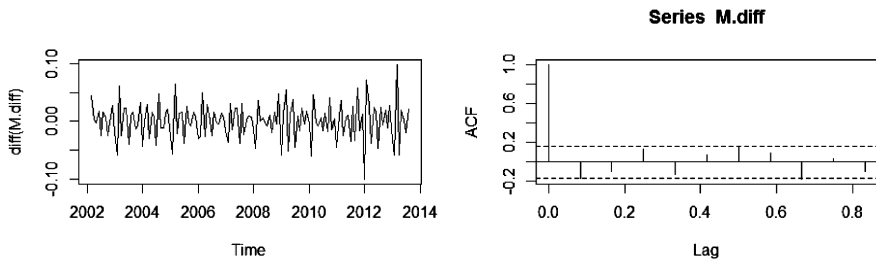


图 13.19 一阶差分序列的时序图和自相关图

取对数并进行一阶差分后，序列在 10 阶内的自相关系数基本上都落在置信区间内，可以认为其是平稳的。另外，对平稳性的检验称为“单位根检验”，在 R 中可以利用程序包 `fUnitRoots` 实现。本例的目的在于预测，因此在此对其不作赘述。

13.4.3 建模

平稳化完成后，要根据时间序列模型的识别规则，对平稳序列建立相应的模型，这里的规则可以总结为：

- ① 若平稳序列的偏相关系数是截尾的，而自相关系数是拖尾的，则序列适合 AR 模型；
- ② 若平稳序列的偏相关系数是拖尾的，而自相关系数是截尾的，则序列适合 MA 模型；
- ③ 若平稳序列的偏相关系数和自相关系数均是拖尾的，则序列适合 ARMA 模型。

也就是说，我们要寻找 $ARIMA(p,d,q)$ 中合适的 p 值和 q 值，一般通过平稳时间序列的自相关图和偏相关图判断。

R 中分别用函数 `acf()` 和函数 `pacf()` 绘制自相关图和偏相关图，它们的调用格式相同，若将参数 `plot` 设为 `FALSE`，将返回自相关和偏相关的真实值，而不再绘图。图 13.19 的右图是一阶差分序列的自相关图，可以看出序列的一阶自相关系数略微超过置信界限，其余各阶自相关均在虚线内，因此可以设置为 p 值等于 1 的自回归。

```
> pacf(M.diff, lag.max=10)
```

绘制结果如图 13.20 所示。

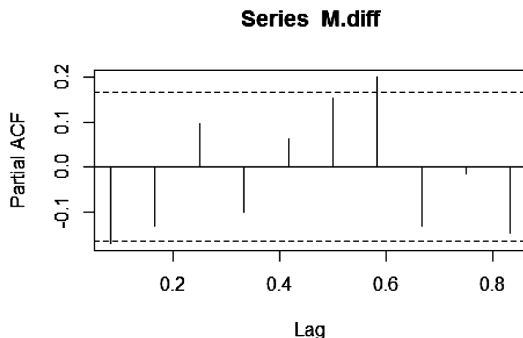


图 13.20 一阶差分的偏自相关图

图 13.20 中滞后 1 阶的偏自相关值超过了置信界限，后面除了 7 阶偏自相关系数较大外，其他各阶的数值都在虚线内，所以初步判断是阶数 $q=1$ 的移动平均。

根据图形判断阶数往往需要主观选择，在没有足够把握的情况下，R 也提供了一个“快捷方式”：通过程序包 `forecast` 中的函数 `auto.arima()` 来发现恰当的 ARIMA 模型，默认 q 和 p 的最大阶数都是 5 阶。还可以使用不同的标准来选择合适的模型，通过参数 `ic` 来设置，通常我们会选择 BIC 作为模型选择标准，它对参数个数的要求非常严格。

```
> library(forecast)
This is forecast 4.06
> auto.arima(M.diff, ic="bic")
Series: M.diff
ARIMA(0,0,0)(0,1,1)[12]

Coefficients:
      sma1
```

```

      -0.7523
s.e.    0.0811

sigma^2 estimated as 0.0002172: log likelihood=350.4
AIC=-696.81  AICc=-696.71  BIC=-691.12

```

根据 BIC 准则, R 自动为我们选择了 ARIMA(0,1,1)的一阶差分、一阶移动平均模型对原始的时间序列建模, 即 $p=0$, $d=1$, $q=1$, 这里 d 是差分所需要的阶数。

13.4.4 模型的参数估计

一旦选择好了时间序列数据的 ARIMA(p, d, q)模型, 使用函数 `arima()` 可以估计出 ARIMA(p, d, q)模型中的参数, 它的调用格式为

```
arima(x, order = c(0, 0, 0), seasonal = list(order = c(0, 0, 0), period = NA), ...)
```

其中, x 是单变量的时间序列; 参数 `order` 按顺序设置 p 、 d 、 q 的值, 是一个数值向量; `seasonal` 设置 ARIMA 模型中的季节性部分, 非季节性的模型则无需设置, 其他参数具体参见 R 帮助文档。

根据自相关图和偏相关图的分析结果, 我们将参数设置为 `order=c(1,1,1)`, 来对货币供应量 M1 的模型作参数估计。注意, 为消除异方差性, 本例中对原始数据取了对数:

```

> M.arima=arima(log(M), order=c(1,1,1))
> M.arima
Series: log(M)
ARIMA(1,1,1)

Coefficients:
      ar1      ma1
      1  -0.992
s.e.      0    0.017

sigma^2 estimated as 0.00042: log likelihood=341.74
AIC=-677.49  AICc=-677.31  BIC=-668.68

```

我们为时间序列选择的模型是 ARIMA(1,1,1), 也就是说我们对一阶差分序列拟合 ARMA(1,1)模型, 用数学形式表示为

$$X_t - \alpha_1 X_{t-1} = \varepsilon_t - \beta_1 \varepsilon_{t-1}, \quad t \in Z$$

上面的参数估计值即 $\alpha_1 = 1, \beta_1 = 0.1054$ 。

13.4.5 模型预测及检验

根据参数估计得到的时间序列模型, 我们可以对未来的序列值进行预测, 并通过程序包 `forecast` 中的函数 `forecast.Arima()` 来完成, 其调用格式为

```
forecast(object, h=10, level=c(80,95),...)
```

`object` 是函数 `arima()` 返回的分析结果对象；`h` 指定预测的未来时点数；`level` 确定预测区间的置信水平，默认情况下将给出 80% 和 95% 置信水平下的预测区间。

例如，我们对未来一年（12 个月）的货币供应量 M1 作预测，由于我们使用的是月度数据，所以参数 `h` 设为 12。

```
> M.pre3=forecast.Arima(M.arima,h=12)
> M.pre3
```

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Sep 2013	12.669	12.642	12.695	12.628	12.709
Oct 2013	12.680	12.642	12.717	12.623	12.737
Nov 2013	12.691	12.645	12.737	12.621	12.761
Dec 2013	12.702	12.649	12.756	12.621	12.784
Jan 2014	12.713	12.654	12.773	12.622	12.805
Feb 2014	12.725	12.659	12.791	12.624	12.826
Mar 2014	12.736	12.664	12.808	12.626	12.845
Apr 2014	12.747	12.670	12.824	12.630	12.865
May 2014	12.758	12.676	12.840	12.633	12.884
Jun 2014	12.770	12.683	12.856	12.637	12.902
Jul 2014	12.781	12.689	12.872	12.641	12.921
Aug 2014	12.792	12.696	12.888	12.645	12.939

仍然用函数 `plot.forecast()` 绘制原始值和预测值，查看图形效果。

```
> plot.forecast(M.pre3)
```

绘制结果如图 13.21 所示。

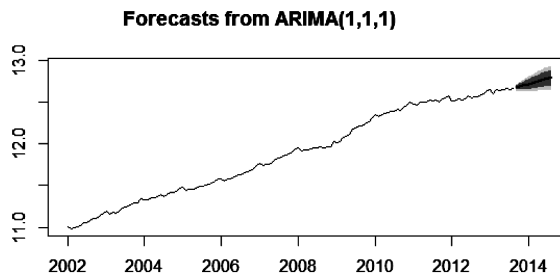


图 13.21 ARIMA 模型预测图

完成 ARIMA 建模的整个过程后，还要对模型进行假设检验，诊断残差序列是否为白噪声。这里的方法与指数平滑法一致，仍采用自相关图和 Ljung-Box 检验结合，来诊断残差序列的自相关性，假设我们查看滞后 5 阶的自相关性：

```
> acf(M.pre3$residuals,lag.max=5)
> Box.test(M.pre3$residuals,lag=5,type="Ljung-Box")
```

Box-Ljung test

data: M.pre3\$residuals

X-squared = 10.699, df = 5, p-value = 0.05769

绘制结果如图 13.22 所示。

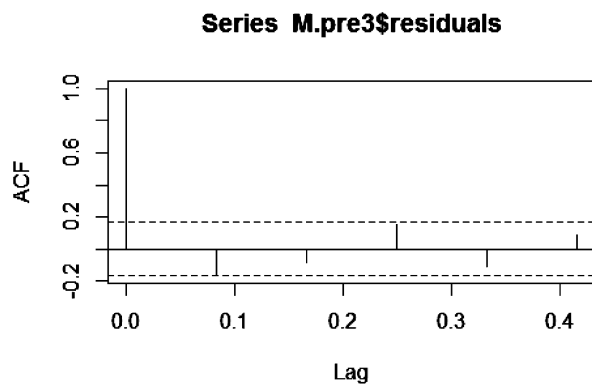


图 13.22 ARIMA 模型的残差自相关图

Ljung-Box 检验的 P 值=0.0577，尽管并不是很显著，但在 0.05 的显著性水平下，我们也没有足够的理由认为可以拒绝原假设。再结合自相关图，滞后 1~5 阶的自相关系数均没有超过置信界限，因此我们认为 ARIMA(1,1,1)模型可以通过白噪声检验，适用于对时间序列 M1 的建模和预测。

下篇

综合实例

➤ 第 14 章 R 在金融数据分析中的应用

➤ 第 15 章 R 在数据预测中的应用

第 14 章

R 在金融数据分析中的应用

数据分析是一项实践性很强的技术，它不可能只停留在理论的层面，更多的是应用于实际，去寻找各个领域关于数字的规律。

把 R 软件与数据分析方法完美地结合，我们才能在实践中，在各式各样的案例分析中使用 R 得到理想的结论。本章将通过实际操作的方法介绍金融数据分析，且选择了较为经典的分析主题，借由 R 语言实现模型构建，并以可视化的图形展现分析结果，向读者展示 R 在金融分析中的应用。

14.1 投资组合最优化实例

14.1.1 概述

证券投资基金是指投资者对各种证券商品进行一定的选择而形成相对固定的若干个投资品种，以达到在一定的约束下，实现投资收益最大化的基本目标。在证券市场上可用于投资的证券种类很多，因此投资者可以建立无数个证券投资基金进行投资，那么何种证券组合才是最有效的投资组合呢？这即为投资组合的最优化问题。

证券投资的目标无非是获得尽可能高的投资收益，并承担尽可能小的风险。为寻求期望收益和投资风险之间的平衡，Markowitz 提出证券组合投资理论，在用横轴表示投资组合的风险 σ_p 、纵轴表示投资组合的预期报酬率 μ_p 的坐标图中，可以求得一条最有效率的投资组合边界曲线 EF，如图 14.1 所示。

本例的目标就是利用 R 软件，根据不同方法寻找投资组合的这一有效边界，绘制出投资机会集的有效边际图像。首先介绍最为经典的马科维茨投资组合理论，马可维茨的风险定价思想在他创建的“均值—标准差”模型中表现得最为清楚。

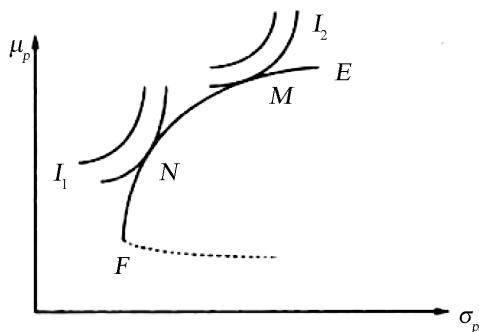


图 14.1 投资组合边界曲线

14.1.2 均值—方差模型

单从现代证券投资组合理论来看，基于不同风险的刻画（如方差、半方差、绝对偏差等度量方式）和不同投资准则（如最大化期望收益率、最小化风险、安全第一准则等）下的证券投资组合理论和模型如雨后春笋般涌出，其中最具有代表性和广为采用的是 M-V 理论、效用理论和随机最优控制理论。

M-V 理论经历了从单阶段到多阶段再到连续时间的发展过程，其优点在于直观，容易被接受，因而在实际中应用非常广泛。但它假设的市场比较简单，不能具体地反映出实际市场的复杂变化情况。

经典的均值—方差（M-V）模型表达式为

$$\begin{aligned} \min f(X) &= X^T V X \\ \text{s.t. } R^T X &= \delta, \sum_{i=1}^n x_i = 1, \text{ 其中, } x_i \geq 0, i=1, 2, \dots, n \end{aligned}$$

n 表示可用于投资组合的证券数目（本文中为 30）， R 为各证券的期望收益向量， X 为各证券投资比例的向量， δ 为投资者的目标收益。

在均值—标准差的二维空间中给出投资机会集的有效边界，单个资产或组合资产的期望收益率由风险测度指标标准差决定：风险越大收益率越高，风险越小收益率越低。风险对收益的决定是非线性（二次）的双曲线（或抛物线）形式，这一结论是基于投资者为风险规避型这一假定而得出的。

均值—方差模型是股票投资组合优化理论中最为朴素的算法，因此在 R 中的实现很容易，简单地讲就是“随机打点法”。我们首先选取投资组合的股票构成，读入数据并作初步的处理。

【数据来源】

上证 50 指数是我国证券市场中具有较大影响力的指数，上证 50 指数成分股是上海证券交易所

所所有股票的一个代表。因此我们选择上证 50 概念股票中的前 30 支股票（按股票代码顺序）在 2012 年 12 月 10 日时点前 220 个交易日的日收益率数据，使用不考虑现金红利的日个股回报率。数据下载于国泰安数据中心。

在 R 中读入原始数据，并计算出 30 支股票的协方差阵和平均收益，以备后续计算使用。输入如下指令：

```
> setwd("d:/data") #设定数据文件所在的路径为默认路径
> data=read.csv("Dalyr.csv",header=TRUE)
> code=unique(data$Stkcd) #提取股票的唯一标识—股票代码
> n=length(code)
> #整理数据
> #计算 30 支股票的协方差阵和平均收益
> Dalyr=matrix(0,220,n)
> for (i in 1:n)
+ {
+   Dalyr[,i]=data$Dretnd[which(data$Stkcd==code[i])]
+ }
> r.cov=cov(Dalyr)#cov 对一个矩阵求协方差,结果为一个矩阵
> r.mean=apply(Dalyr,2,mean) #对每列计算均值
> options(digits=2)
> r.mean
[1] -0.00054-0.00059 -0.00102 0.00016 -0.00068 -0.00015 -0.00067 -0.00156 0.00015
[10] -0.00115 -0.00056 0.00174 -0.00146 -0.00183 0.00028 0.00247 -0.00143 -0.00189
[19] -0.00086-0.00038 0.00074 0.00057 0.00083 0.00037 -0.00264 0.00010 -0.00113
[28] 0.00056-0.00017 0.00056
```

以上为 30 支股票在这 220 个交易日的平均收益率，将其绘成散点图（如图 14.2 所示）。

```
> plot(r.mean,main='30 支股票期望收益率',pch=8,col=2)
```

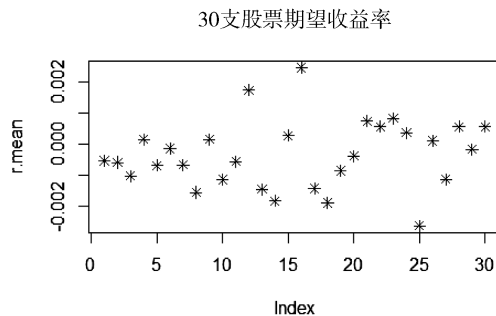


图 14.2 30 支股票的期望收益率

使用朴素方法画出股票投资组合的有效边界，完全依据均值—方差模型的原理，随机模拟 100 000 种投资组合的权重，将所有投资组合的点画在均值方差图中，选取每个期望收益下风险最小

的点连成有效边界曲线。

在 R 中输入如下指令：

```
> ###均值-方差模型###
> t11=Sys.time()
> num=100000#循环十万次，画出随机点
> rp=numeric(num);sigmap=numeric(num) #定义两个变量，存放投资组合的均值和标准差
> for (i in 1:num)
+ {
+   x1=runif(30)
+   x=x1/sum(x1)
+   rp[i]=sum(r.mean*x)
+   sigmap[i]=t(x)%*%r.cov%*%x
+ }
> plot(sigmap,rp,pch='.',main='随机打点法有效边界') #绘制 10 万次模拟的投资组合散点
> #计算有效边缘
> rp=round(rp,4)
> rp.uni=unique(rp)
> nn=length(rp.uni)
> sigma.min=numeric(nn)
> for (i in 1:nn)
+ {
+   sigma.min[i]=min(sigmap[which(rp==rp.uni[i])])
+ }
> rp.sort=sort(rp.uni)
> order=order(rp.uni)
> lines(rp.sort~sigma.min[order],col='red')
> t12=Sys.time()
> time.used1=t12-t11;time.used1#朴素法使用的时间
Time difference of 11 secs
```

绘制结果如图 14.3 所示。

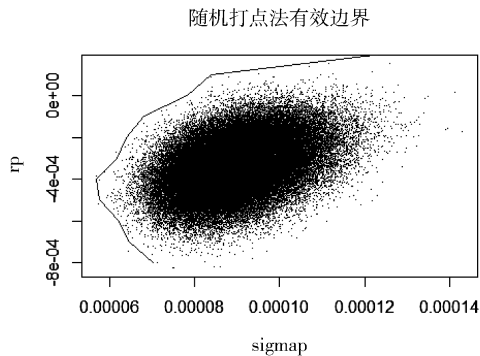


图 14.3 均值—方差法的有效边界

对现有的 30 支股票，随机模拟出 10 万种投资组合的结构，从图 14.3 中散点的密集程度可以发现模拟几乎可以覆盖各种可能的组合，对收益率相等的投资组合寻找标准差最小的那个点，最后就可以连接成有效边界曲线。

但是这种方法毕竟过于朴素，需要大量的模拟才能得到一个相对有效的结果，实现起来需要花费 11 秒的时间。但现实的投资组合远不止 30 支股票，我们需要找到更好的组合最优化算法。

14.1.3 模拟退火算法

均值一方差模型在原理上是一个精确算法，但若要在软件中实现，也要辅助随机模拟，从而变成了一种近似方法。随机模拟中的很多优化算法就是这样一种近似方法，它们不可能完全精确，但却是可以实现的快速算法。接下来我们使用其中的一种——模拟退火算法，来寻找投资组合的有效边界。

【算法原理】

模拟退火算法最早的思想于 1953 年由 Metropolis 等提出，它模拟物理的退火过程，应用于组合最优化，克服了优化过程中局部极小和初值依赖性的缺陷。

（1）物理退火过程

加热过程——使目标组合处于完全随机状态；

等温过程——在某一温度下，系统处于温度不变的封闭系统，系统状态的变化朝自由能减少的方向进行；

冷却过程——温度减小，系统的能量逐渐下降并逐渐趋近有序，从而得到最优的状态。

模拟退火算法将上述过程应用于组合最优化问题，将内能模拟为组合的目标函数值，两者相似性如表 14.1 所示。

表 14.1 模拟退火法与物理退火过程

组合优化问题	金属物体
解	粒子状态
最优解	能量最低的状态
设定初温	熔解过程
Metropolis 抽样过程	等温过程
控制参数的下降	冷却
目标函数	能量

（2）Metropolis 抽样过程

实现组合在某一温度下达到最优状态的过程。应用 Metropolis 准则并适当地控制温度的下降过程实现模拟退火，从而达到求解全局优化问题的目的。这一过程使用 Monte Carlo 加以模拟。

模拟退火算法对应了一个马尔可夫链, 新状态接受概率仅依赖于新状态和当前状态, 这是 Metropolis 抽样的基础。在温度 T 下, 组合停留在某一状态满足 Boltzmann 概率分布, 因此有

$$P\{\bar{E} = E_1\} - P\{\bar{E} = E_2\} = \frac{1}{Z(T)} \exp\left(-\frac{E_1}{T}\right) \left[1 - \exp\left(-\frac{E_2 - E_1}{T}\right)\right]$$

若在温度 T 下, 选取当前状态 i 的相邻状态 j , 若 $E_j < E_i$, 则接受 j 为当前状态; 否则, 若概率 $p = \exp[-(E_j - E_i)/T]$ 大于 $[0,1]$ 区间的随机数, 则仍接受状态 j 为当前状态; 若前面两个条件均不成立, 则保留状态 i 为当前状态。

【投资组合优化的模拟退火过程】

根据 Metropolis 准则, 将内能 E_i 模拟为目标函数值 f , 温度 T 演化成控制参数 T , “解”即为最优投资组合的权重值, 即得到解投资组合优化问题的模拟退火算法: 由初始解 f_0 和控制参数初值 T_0 开始, 对当前解重复“产生新解-计算目标函数差-判断是否接受-接受或舍弃”的迭代, 并逐步衰减 T 值, 算法终止时的当前解即为所得近似最优解。

(1) 思路

给定初温 $T=T_0$, 随机产生初始状态 $f=f_0$, 令 $k=1$;

```
Repeat
  Repeat
    产生新状态  $f_j = \text{Generate}(f)$ ;
    If  $\min\{1, \exp[-(f_j - f)/T_k]\} > \text{random}[0,1]$   $f = f_j$ ;
  Until 抽样稳定准则满足;
  退温  $T_{k+1} = \text{update}(T_k)$  并令  $k=k+1$ ;
Until 算法终止准则满足;
输出算法搜索结果。
```

(2) 模拟退火法的步骤

1. 初始化。给定初始温度 $T = T_0$, 随机产生初始状态 $X = X_0$, 确定在每个温度下的迭代次数 L ;
2. 在温度 T 下对 $k=1, 2, \dots, L$ 重复步骤 3 和步骤 6;
3. 产生相邻状态 X ;
4. 计算评价目标函数的增量 $\Delta f = f(X') - f(X)$;
5. 如果 $\Delta f < 0$, 则接受新解; 否则, 如果 $\exp(-\Delta f / T)$ 大于 $[0,1]$ 上的一个随机数, 则接受新解 (设置双阈值);
6. 如果满足终止条件则输出当前解作为最优解, 结束计算;

7. 退火，然后转至步骤 2。

(3) 目标函数的选择

根据 M-V 模型，投资组合的最优化是一个多约束条件下的优化问题，这在模拟退火算法中是首先要解决的问题。为简化计算的思路，可将多个约束化为一个条件，用如下方法建立目标函数。

方法 1：乘法函数法

应用乘法函数法可以将 M-V 模型转化为如下的惩罚函数 $L(X)$ 的无约束优化问题来进行求解：

$$\min L(x) = f(X) + M(\max(0, (\delta - R^T X)))$$

其中， M 为惩罚因子，是充分大的正数。

方法 2：正态假设法

以 δ 为目标收益率，那么投资组合最优化的目标即使组合收益率达到 δ 的概率最大化 $\max_x P(r_p \geq \delta)$ 。假设每个证券的期望收益率均服从正态分布，那么投资组合的收益率也服从正态分布，即

$$r_p = \sum_{i=1}^n x_i r_i \sim N(R^T X, X^T \Sigma X)$$

从而概率最大化的模型可以表示为

$$\min_x f(x) = \min_x \left[\frac{\delta - R^T X}{(X^T \Sigma X)^{1/2}} \right]$$

(4) 参数选择

1) 目标收益率 δ 。 δ 应大于所选取的 30 支股票的最高收益率，本文选择 0.0025。

2) 相邻状态的产生

由于模拟退火算法的 Markov 性，相邻状态的产生依赖于当前状态。假设当前状态下投资组合权重为 $X^k = (x_1, x_2, \dots, x_n)$ ，它的相邻状态产生函数为 $x_i^{k+1} = x_i^k + \alpha(b - a) = x_i^k + \alpha$ 。其中， α 为 (0,1) 上的随机数， $[a, b]$ 为 x 的取值范围，即 [0,1]。

3) 温度衰减函数。 $T_k = \frac{T_0}{k^m}$, $k=1, 2, \dots$ 其中， T_0 是初始温度， m 是一个大于等于 1 的常数，通常取 3； T_k 是第 k 次降温后的温度。

4) 终止条件。温度最终降为 $T=0.0001$ （接近零度，冷却状态）时循环过程结束。

在参数选择的过程中,对初始温度及内层迭代的次数没有进行设置,由于这两个参数直接决定着算法的循环次数以及优化效果,初始温度的选择越高,则退温次数越多,搜索到全局最优的可能性越大,从而得到的结果比较稳定。但算法的迭代次数增加会降低算法的可行性和有效性,因此本文通过几次尝试,选择最适合的参数来控制循环次数和结果。

首先使用乘法函数法的目标函数,设置初始温度为 10 000,内循环次数为 10,实现的代码如下:

```
> ###simulated annealing###
> t2l=Sys.time()
> temp0=10000 #确定初温
> temp=temp0
> x0=c(1,runif(29))
> x=x0/sum(x0)
> L=10 #循环次数
> M=1000 #惩罚因子
> R=0.0025 #目标收益率
> f1=function(x){t(x)%*%r.cov%*%x+M*max(0,R-sum(r.mean*x))} #乘法函数法
> f2=function(x){(R-sum(r.mean*x))/sqrt(t(x)%*%r.cov%*%x)} #正态假设法

> f=f1(x) #记录每次退温的最优解
> f=as.numeric(f)
> k=1 #记录退温次数
> a=data.frame(x)#记录权重
> e=0.00001
> while(temp>0.001)
+ {
+   for(i in 1:L)
+   {
+     x1=x+runif(30)
+     x1=x1/sum(x1)
+     deltaf=f1(x1)-f1(x)
+     pr=exp(-deltaf/temp) #Meropolis 判断概率
+     if(deltaf<0){x=x1;f[k+1]=f1(x1)} else {if(pr>runif(1)) x=x1;f[k+1]=f1(x1)}
+   }
+   k=k+1
+   a[k]=x
+   temp=temp0/(k^3)#退温
+ }

> #退火法选取的投资组合
> r.sa=0;sigma.sa=0
> for (i in 1:length(a))
+ {
+   r.sa[i]=sum(r.mean*a[[i]])
```

```

+   sigma.sa[i]=t(a[[i]])%*%r.cov%*%a[[i]]
+ }
> plot(sigma.sa,r.sa,main='模拟退火法有效边界')
> #有效边缘
> r.sa=round(r.sa,4)
> rsa.uni=unique(r.sa)
> nn=length(rsa.uni)
> sigma.min=numeric(nn)
> for (i in 1:nn)
+ {
+   sigma.min[i]=min(sigma.sa[which(r.sa==rsa.uni[i])])
+ }
> rsa.sort=sort(rsa.uni)
> order=order(rsa.uni)
> lines(rsa.sort~sigma.min[order],col='red',lwd=2)
> legend(0.000101,-4e-04,legend='正态假设法')
> t22=Sys.time()
> time.used2=t22-t21;time.used2#退火法使用的时间
Time difference of 1.3 secs

```

很明显,模拟退火算法大大提高了计算效率,其仅使用了 1.3 秒就完成了所有的步骤,绘制的有效边界图如图 14.4 所示。由于循环次数过少使得模拟的有效边界不完整,且投资组合的点过于分散。

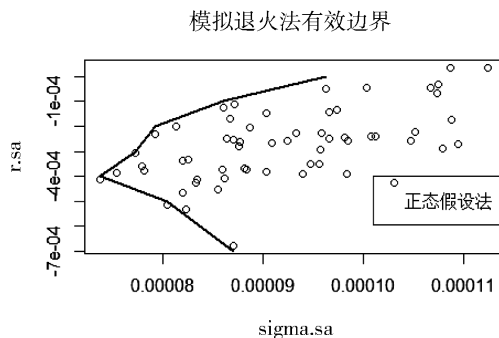


图 14.4 模拟退火法的有效边界 (初始温度为 10 000, 内循环次数为 10)

当我们把初始温度增加至 100 000, 内循环次数增为 50 时, 共退温 465 次, 优化效果得到明显的改善, 因此选取 $T_0 = 100\,000$, $L = 50$ 。有效边界如图 14.5 所示。

花费的计算时间为 4.9 秒。

```

> time.used2=t22-t21;time.used2#退火法使用的时间
Time difference of 4.9 secs

```

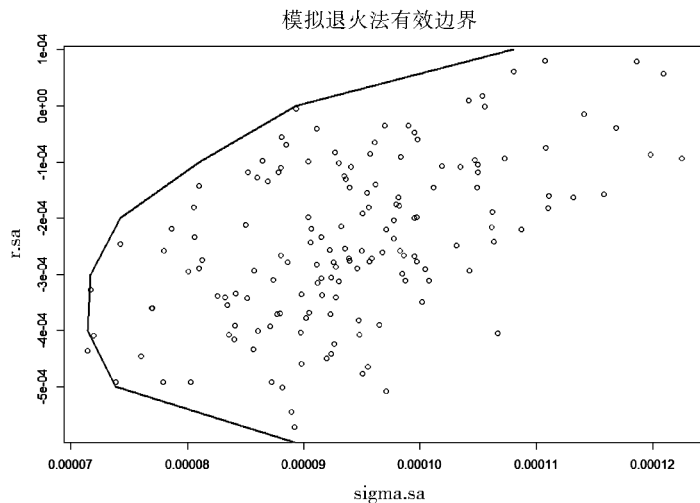


图 14.5 模拟退火法的有效边界（初始温度为 100 000，内循环次数为 50）

使用正态假设下的目标函数（ f_2 ），同样也可算出有效边界（见图 14.6）。

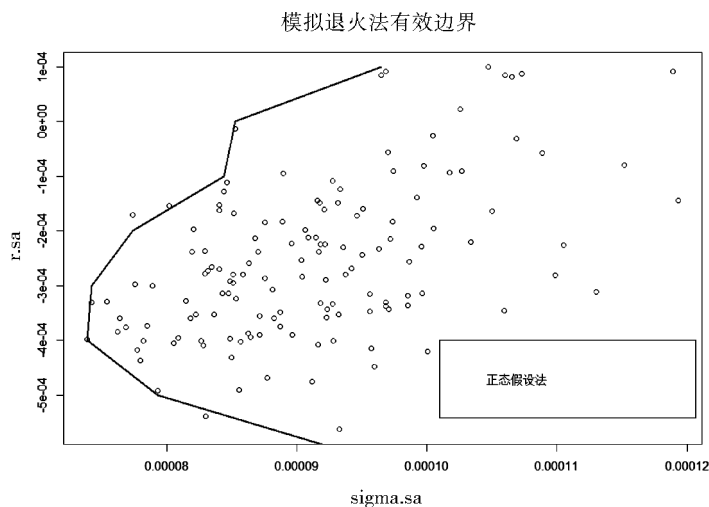


图 14.6 模拟退火法的有效边界（正态假设）

与朴素法比较，模拟退火法明显改善了计算有效边界的效率，通过筛选条件选取接近有效边界的点，算法明显加快。

使用 `Sys.time()` 函数计算，朴素方法使用的时间为 11 秒，而初温为 100 000、内层迭代次数为 50 的模拟退火方法用的时间为 4.9 秒（乘法函数法目标函数）、5 秒（正态假设下的目标函数），效率提高了，且得到了更好的有效边界图形。而目标函数的选择对于结果和效率影响并不大。

注意：由于 Monte Carlo 随机模拟的随机性，因此每次计算的图形结果和使用时间有一定出入，文中选取了较好的图形效果展示。

14.2 构造投资组合的有效前沿

14.2.1 R 中的算法包

构造投资组合的有效前沿一直都是金融投资分析中的重要问题，前面介绍的均值—方差模型是股票投资组合优化理论中最为朴素的算法。

随着 R 使用者的不断研究开发，R 目前已经拥有了可以用于计算有效前沿的函数，其比起随机模拟计算将更为精确，也更加简单。在 R 中实现投资组合有效前沿的计算，依靠以下三个步骤完成：

1. 读入数据；
2. 将数据加工处理，得到收益率矩阵；
3. 以收益率矩阵作为输入变量，计算得到有效前沿。

在获取数据之前，我们首先介绍一个在金融分析中常用的程序包 `quantmod`，它是 Quantitative Financial Modeling & Trading Framework 的简写，顾名思义其是进行金融定量建模和交易分析的程序包，作者为 Jeffrey A. Ryan。

`quantmod` 中提供了获取金融数据的函数 `getSymbols()`，它在获取数据方面功能很强大且使用起来非常便捷，可以直接从网站上下载数据，其调用格式为

```
getSymbols(Symbols = NULL, env = parent.frame(), reload.Symbols = FALSE,
  verbose = FALSE, warnings = TRUE, src = "yahoo", symbol.lookup = TRUE,
  auto.assign = getOption('getSymbols.auto.assign', TRUE), ...)
```

`Symbols` 是一个字符向量，用于指定要加载的数据名称，一般为上市公司的名称，例如要获取 Google 公司的数据，则输入“GOOG”，要得到中国移动的数据，则输入“CHL”；`env` 指定创建对象的位置；`verbose` 是逻辑值，指定是否返回检索状态；`warnings` 也是一个逻辑值，表示是否返回警告；`src` 是除 `Symbols` 外最为重要的参数，它是一个字符向量，指定数据来源，默认为“yahoo”，若要从 Google 上获取数据，则改为“google”即可。

14.2.2 计算分析

接下来我们就可以从网站下载数据了。由于分析目的是构建资产组合的有效前沿，所以我们需要同时获取多支股票的交易数据，为简化计算，这里以三支股票：IBM、SPY、YHOO 为例。

```
> install.packages("quantmod")
> library(quantmod) #载入程序包
```

```
> getSymbols(c('IBM','SPY','YHOO')) #获取数据
[1] "IBM" "SPY" "YHOO"
```

这样导入的数据结构非常完整,包括交易的日期(2007-01-03 至当前时点)、开盘价、最高价、最低价、收盘价、交易量和调整收盘价 7 个变量。

第二步,计算收益率矩阵。由于我们获得的是日交易数据,因此需要计算日收益率,这一步通过函数 `dailyReturn()` 即可完成,并通过函数 `merge()` 将三支股票收益率序列合并为矩阵。程序包 `quantmod` 中包含计算不同期限收益率的函数,如 `weeklyReturn`、`monthlyReturn`、`quarterlyReturn` 和 `annualReturn` 等。

```
> IBM_ret=dailyReturn(IBM)
> SPY_ret=dailyReturn(SPY)
> YHOO_ret=dailyReturn(YHOO)
> data=merge(IBM_ret,SPY_ret,YHOO_ret) #合并收益率
```

第三步,计算投资组合的有效前沿。这里必须用到程序包 `fPortfolio` 中的函数 `portfolioFrontier()`,它的输入参数必须是时间序列对象,即 `timeSeries` 类,因此使用之前应该先将数据格式转换为时序对象,可通过函数 `as.timeSeries()` 完成转换,它存在于程序包 `timeSeries` 中。

```
> library(timeSeries)
> data=as.timeSeries(data) #转换对象
> install.packages("fPortfolio")
> library(fPortfolio)
> Frontier=portfolioFrontier(data) #计算有效前沿
> Frontier
```

```
Title:
MV Portfolio Frontier
Estimator:      covEstimator
Solver:         solveRquadprog
Optimize:       minRisk
Constraints:     LongOnly
Portfolio Points: 5 of 50
```

```
Portfolio Weights:
      daily.returns  daily.returns.1  daily.returns.2
1          0.0000          1.0000          0.0000
13         0.3713          0.6287          0.0000
25         0.6465          0.2901          0.0635
37         0.7791          0.0000          0.2209
50         0.0000          0.0000          1.0000
```

```
Covariance Risk Budgets:
      daily.returns  daily.returns.1  daily.returns.2
1          0.0000          1.0000          0.0000
```

13	0.3569	0.6431	0.0000
25	0.6658	0.2650	0.0692
37	0.7169	0.0000	0.2831
50	0.0000	0.0000	1.0000

Target Return and Risks:

	mean	mu	Cov	Sigma	CVaR	VaR
1	0.0003	0.0003	0.0146	0.0146	0.0358	0.0226
13	0.0003	0.0003	0.0137	0.0137	0.0331	0.0209
25	0.0004	0.0004	0.0139	0.0139	0.0333	0.0218
37	0.0005	0.0005	0.0150	0.0150	0.0353	0.0233
50	0.0006	0.0006	0.0273	0.0273	0.0584	0.0360

Description:

Tue Feb 18 23:16:10 2014 by user: Administrator

计算的结果分为多个部分，首先 **title** 部分给出均值一方差有效前沿计算过程中涉及的相关方法。**Portfolio Weights** 部分返回的是三支股票在投资组合中的头寸比例，第一列的数字表示从绘图中的 50 个点中选取有代表性的 5 个点，每行表示在该点上股票在组合中的比例，行和都是 1。

例如，对于第三行，它表示的是第 25 个点在投资组合中将总头寸以 64.65%、29.01%、6.35% 的比例分散到三支股票标的上。**Covariance Risk Budgets** 表示各点的协方差风险预算矩阵。**Target Return and Risks** 表示目标组合的预期收益率和风险数据，也就是有效前沿绘图的依据。

计算出 **Frontier** 后，可以调用函数 **plot()** 直接绘图。输入如下指令后，R 会返回一组绘图选项，最后以 “**selection:**” 结尾，这时我们可以输入相应的图形编号，从而选择不同的绘图结果。

```
> plot(Frontier)
```

Make a plot selection (or 0 to exit):

```
1: Plot Efficient Frontier
2: Add Minimum Risk Portfolio
3: Add Tangency Portfolio
4: Add Risk/Return of Single Assets
5: Add Equal Weights Portfolio
6: Add Two Asset Frontiers [LongOnly Only]
7: Add Monte Carlo Portfolios
8: Add Sharpe Ratio [Markowitz PF Only]
Selection:
```

关于图形编号，输入 “1” 直接绘出有效前沿，2~8 表示在有效前沿的图形中添加绘图结果，其含义依次是：最小风险组合、切线组合、单个资产的风险/收益、等权重投资组合、两资产投资组合的有效前沿（禁止卖空）、蒙特卡洛模拟得到的投资组合、夏普比率（仅马克维茨有效前沿）。

依次输入 1~8（除去 7，随机模拟得到的散点），可以得到如图 14.7 所示的完整图形。

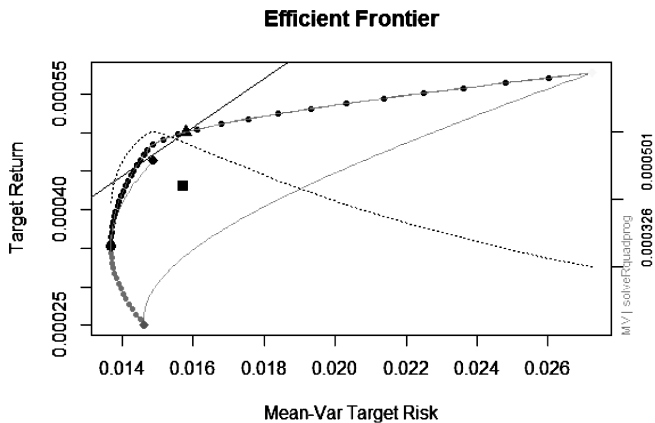


图 14.7 投资组合有效前沿

绘图完成后，输入“0”或按 Esc 键即可退出。相比随机模拟方法，直接调用 R 软件中的专业程序包可以更快速、精确地完成投资组合有效前沿的计算和绘图。

14.3 股票聚类分析

14.3.1 概述

上一个案例是股票投资组合的最优化，其针对已有的股票组合寻找它们的有效边界。但是目前我国股票市场上有几千支股票，应该如何选择是一个投资组合呢？所以在进行股票投资之前，我们首先要对股票进行分析和选择，而股票评价指标有很多，反映的侧重点各不相同，如何把这些指标综合在一起对股票进行分类就变得十分重要。

“板块”这一概念是指具有共同特征的股票群，股市中的板块可以从多个角度来划分，如行业、产业、地域等，在本例中我们选取了钢铁行业的股票为分析对象。在每个板块中，有几十种甚至上百种股票，每支股票背后又对应着各公司众多的财务指标，面对这么复杂的数据，如何才能找出具有相同特点的股票呢？

聚类分析是数据挖掘中的核心技术，它是通过数据建模简化数据的一种方法，第 12 章我们已经介绍过 R 软件中的聚类分析函数。本例中我们将聚类分析方法应用到钢铁行业股票的考察中去。

选取钢铁行业的 20 支股票及 7 个指标作为聚类分析的原始依据，数据总结如表 14.2 所示。

表 14.2 钢铁行业股票的原始数据

股票名称	总资产 (亿元)	主营收入 (亿元)	净利润 增长率	每股净资产 (元)	净资产 收益率	主营业务收 入增长率	每股资本 公积金 (元)
大钢不锈	452.23	178.65	759.48	5.463	8.43	210.12	2.2008
安阳钢铁	159.27	49.33	721.72	3.38	2.94	41.78	0.387
鲁银投资	15.67	8.44	649.82	1.47	2.13	36.15	0.144
南钢股份	100.09	50.77	614	4.03	6.24	63.67	1.2055
武钢股份	490.85	124.1	604.6	2.99	6.026	49.03	0.7368
莱钢股份	159.96	69.38	549.15	6.21	3.86	21.54	1.477
柳钢股份	102.64	46.78	456.77	6.095	4.97	45.11	1.4646
凌钢股份	34.92	16.01	331.54	5.27	3.15	12.9	0.8365
华凌股份	389.38	92.63	330.87	4.71	2.64	31.52	1.8373
济南钢铁	160.11	76.93	325.84	4.25	6.18	37.9	1.4288
唐钢股份	291.07	95.69	313.61	4.3	4.77	54.34	1.4273
杭钢股份	91	32	308.82	5.18	2.09	17.98	1.1401
安泰股份	37.53	7.92	250.86	3.28	2.73	44.13	0.8879
承德钒钛	108.6	30.23	221.52	3.31	1.79	58.59	1.162
韶钢松山	128.36	32.16	213.71	4.07	1.76	9.79	1.0413
本钢板材	259.5	77.03	207.04	5.2323	1.92	89.47	2.7758
八一钢铁	92.24	26.68	185.25	4.19	1.6	71.53	1.4495
宝钢股份	1778.4	429.75	156.38	4.85	4.32	28.66	1.8749
鹏博士	5.93	1.6	153.05	1.471	0.93	9.57	0.329
广钢股份	44.14	12.87	145.41	2.13	0.92	43.53	0.8744

14.3.2 K-means 聚类分析

K-means 聚类分析是一种快速聚类方法，适合处理大样本数据。K-means 聚类分析要求变量为数值型，其基本思想是使聚类性能指标最小化，聚类准则函数计算数据集中每个样本点到该分类中心的距离平方和。我们需要事先指定分类数 n ，各分类中心的初值可以自行设置，也可以由程序自动给出。

K-means 聚类分析采用迭代算法，首先根据分类数 n ，为每个类别确定一个类中心初值；然后按最小距离原则将各样本分配到邻近的类别中，得到每个类别的样本均值作为新的类中心；如此反复，不断调整各分类中心的位置，直到收敛，最终可以得到 n 个类别。

R 软件中实现 K-means 方法的函数是 `kmeans()`，它的调用格式为

```
kmeans(x, centers, iter.max = 10, nstart = 1,
algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"))
```


其中, x 是待研究的数据矩阵; **centers** 是指定分类个数的数值或指定聚类中心的初值; 如果 **centers** 指定了聚类个数, 则 **nstart** 表示选取的随机集个数; **iter.max** 是最大迭代数, 默认为 10; **algorithm** 表示具体的算法, 默认为 Hartigan-Wong 方法。

接下来读取数据, 并用 K-means 聚类方法把上面的 20 支股票分为 4 组。

```
> stock=read.table("d:/data/stock.txt",header=T)
> rownames(stock)=stock[,1]
> KM=kmeans(stock[,2:8],4)
> KM
K-means clustering with 4 clusters of sizes 5, 2, 1, 12

Cluster means:
      A1      A2      A3      A4      A5      A6      A7
1 229.6783 80.11167 649.795 3.923833 4.937667 70.38167 1.0251833
2 240.5400 77.81200 326.826 4.917460 4.096000 51.66800 1.7867600
3 778.4000 29.75000 156.380 4.850000 4.320000 28.66000 1.8749000
4  67.8400 19.93375 226.270 3.612625 1.871250 33.50250 0.9650875

Clustering vector:

大钢不锈 安阳钢铁 鲁银投资 南钢股份 武钢股份 莱钢股份 柳钢股份 凌钢股份
      2      1      1      1      2      1      1      4

华凌股份 济南钢铁 唐钢股份 杭钢股份 安泰股份 承德钒钛 韶钢松山 本钢板材
      4      4      4      4      4      4      4      4

八一钢铁 宝钢股份 鹏博士 广钢股份
      4      3      4      4

Within cluster sum of squares by cluster:
[1] 57468.91 27209.52      0.00 229013.29
(between_SS / total_SS = 91.8 %)

Available components:
[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size" > KM=kmeans(stock[,2:8],4)
```

分析结果中的 **Clustering vector** 给出了每支股票对应的类别, 用函数 **sort()** 对分类情况排序, 并整理得到结果:

```
> sort(KM$cluster)

安阳钢铁 鲁银投资 南钢股份 莱钢股份 柳钢股份 大钢不锈 武钢股份 宝钢股份
      1      1      1      1      1      2      2      3
```

凌钢股份	华凌股份	济南钢铁	唐钢股份	杭钢股份	安泰股份	承德钒钛	韶钢松山
4	4	4	4	4	4	4	4
本钢板材	八一钢铁	鹏博士	广钢股份				
4	4	4	4				

14.3.3 层次聚类分析

基于层次的聚类方法在第 12 章已经介绍过，其基本思想是，初始将各个样本各自作为一类，确定好计算样本之间距离和类之间距离的方法，然后把最近的两类（两个点）合并成一类，再把剩下的最近的两类合并成一类；重复合并距离最近的两个类，这样下去，每次都减少一类，直到最后只有一个大类为止。自下而上地，最终形成了层次关系。

选择欧式距离作为计算样本之间的距离，并基于 Ward 方法进行系统聚类，在 R 中用函数 `hclust()` 实现具体过程，并绘制谱系图。与 Kmeans 方法一样，我们最后仍把 20 支股票分为 4 类。在 R 中输入如下指令：

```
> subset=subset(stock,select=2:8) #去掉第一列的股票名称
> d=dist(subset) #计算欧式距离
> hc=hclust(d,method="ward") #进行Ward聚类
> plclust(hc) #绘制分层聚类的谱系图
> rect.hclust(hc,k=4,border="red")
```

绘制结果如图 14.8 所示。

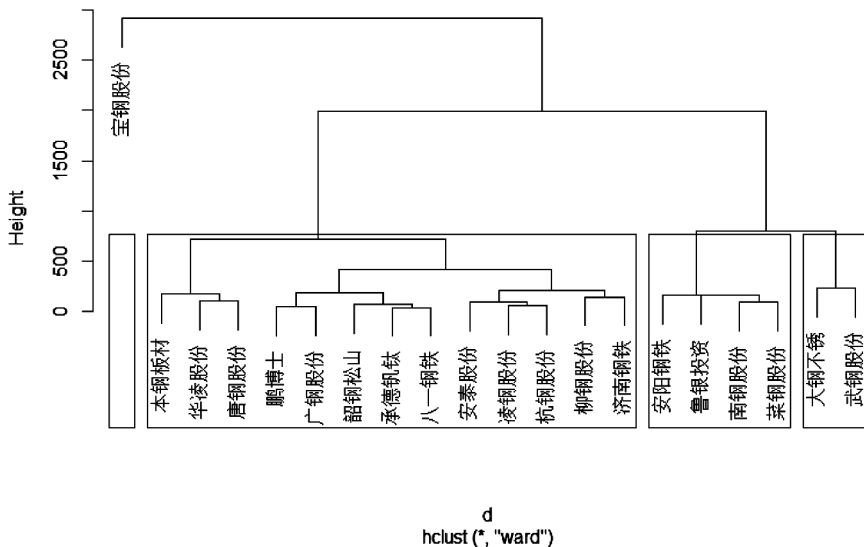


图 14.8 分层聚类的谱系图

聚类谱系图直观地显示了聚类的过程，从图 14.8 中可以清楚地看出各种股票的归属。可以发现，运用分层聚类方法对股票分类的结果与 K-means 聚类方法十分一致，唯一不同的仅仅是“柳钢股份”的类别。

每个类别的股票都具有其各自的特征，结合它们的财务指标，我们可以总结出：第一类的“宝钢股份”是中国总资产规模最大、最现代化的钢铁联合企业，是比较传统的钢铁公司，其综合竞争力很强。对于这样的股票，它无论是涨还是跌，幅度都不会很大；第三类股票总体来说处于均衡位置，有一定的投资效果，但不明显；最后一类属于潜力型的股票，未来前景一片大好。

聚类分析能综合多项财务指标来反映上市公司的盈利能力和发展状态，从而帮助我们根据多个特征对股票分类和评估，在此基础上进行股票投资组合筛选，会更加容易。

第 15 章

R 在数据预测中的应用

预测分析是很多行业都非常需要的一项方法和技术，例如经济走势需要预测 GDP 和 CPI，气象需要预测未来几天的天气变化，农业需要预测农作物的长势和产量，甚至篮球比赛也会预测对战双方的胜负情况。

随着数据积累越来越多，如何使用统计软件实现对数据的深入挖掘和预测成为许多行业面临的问题。数据预测主要有两个方面：回归分析预测和时间序列预测，本章就这两部分来分别举实例说明 R 在预测方面的应用。

15.1 回归分析预测

15.1.1 概述

回归分析预测法，是在分析自变量和因变量之间相关关系的基础上，建立变量之间的回归方程，并将回归方程作为预测模型，根据自变量在预测期的数量变化来预测因变量，因此，回归分析预测法是一种重要的预测方法。当我们对未来发展状况和水平进行预测时，如果能将影响预测对象的主要因素找到，并且能够取得其数量资料，就可以采用回归分析预测法进行预测。它是一种具体的、行之有效的、实用价值很高的常用市场预测方法。

回归分析预测法有多种类型。依据相关关系中自变量的个数不同分类，可分为一元回归分析预测法和多元回归分析预测法。在一元回归分析预测法中，自变量只有一个；而在多元回归分析预测法中，自变量有两个以上。依据自变量和因变量之间的相关关系不同，可分为线性回归预测和非线性回归预测。

15.1.2 实战案例

人口老龄化是几乎全球各国都面临的一个现实问题，随着人类生活水平的提高、医疗技术的

进步，人类的寿命不断延长，各国的人口死亡率逐年下降。社会的老龄化会带来一系列问题，影响一国的经济、政策和发展等方方面面，因此死亡率的准确预测对一个国家而言十分重要。

在第 4 章中，我们曾经绘制过一个瑞典死亡率数据的三维图，这一数据集来自人类死亡率数据库 (HMD,2007)，提供了 1951–2005 年瑞典的人口信息。这是一个多变量的数据集，变量的描述如表 15.1 所示。我们的研究目的是拟合瑞典的死亡率模型，选取其中的男性人口数据，寻找死亡率与年份、年龄之间的关系，从而根据拟合的模型对未来做出预测。

表 15.1 瑞典死亡率数据描述

变量名称	描述
Year	年份: 1951,1952,...,2005
Age	年龄: 0,1,...,109
Male_Exp	男性生存人口数
q_male	男性死亡率
Male_death	男性死亡人数
L_female_exp	对数男性生存人口数

研究多变量关系的统计模型有很多，死亡率模型也有很多，本例中我们选择前面介绍过的广义线性模型，模拟瑞典的死亡率与年龄和年份之间的关系。一个完整的数据分析过程包括很多步骤，模型是其主干，但初步的描述、探索性分析以及最后的结论都是必不可少的部分。

(1) 数据的处理和描述

首先要说明的一点是，实际分析中得到的数据往往是未经过处理的，所以可能含有缺失值、异常值等诸多问题数据，所以读入数据后的第一步就是进行一定的处理。

在瑞典死亡率的数据集中，共包含 6000 多条记录，在缺失值相对不算太多的情况下，剔除是最快捷的处理办法，使用函数 `na.omit()` 完成。对于死亡率数据而言，有一个特殊点是取值范围 $0 < q \leq 1$ ，不在此范围内的点均视为异常值，也应当剔除，否则会影响拟合效果。

于是，在 R 中输入如下指令读取并预处理数据：

```
> setwd("d:/data") #设置文件读取的路径
> mortality=read.csv('swedish mortality.csv',header=T)
> mortality=na.omit(mortality)
> mortality=mortality[mortality$q_male>0,]
> mortality=mortality[mortality$q_male<=1,]
> attach(mortality)
```

整理好数据之后，绘制散点图是查看待研究变量之间关系的最直接方式，本例构建广义线性模型涉及多个变量，可以分别绘制三维和二维的散点图来具体分析。

三维图要用到程序包 `rgl` 中的函数 `plot3d()`，绘制的三维散点图可以拖动鼠标旋转，以得到合

适的观察图像的角度。

```
> library(rgl)
> plot3d(Year, Age, q_male, col='grey', type='p', zlim=1)
> #二维散点图
> par(mfrow=c(1,2))
> plot(Age, log(q_male), main='年龄与死亡率 (对数)')
> plot(Year, log(q_male), main='年份与死亡率 (对数)')
> layout(1) #取消图形区域拆分
> plot(Age, L_male_exp, main='年龄与对数生存人数')
```

绘制图形分别如图 15.1、图 15.2 及图 15.3 所示。

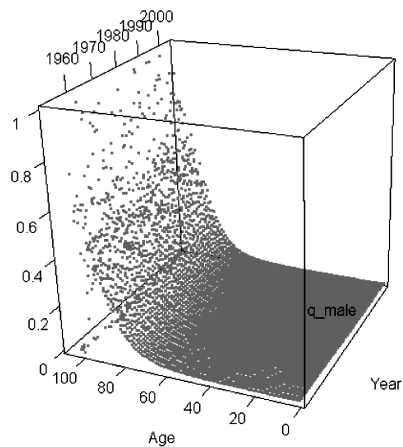


图 15.1 变量关系——三维散点图

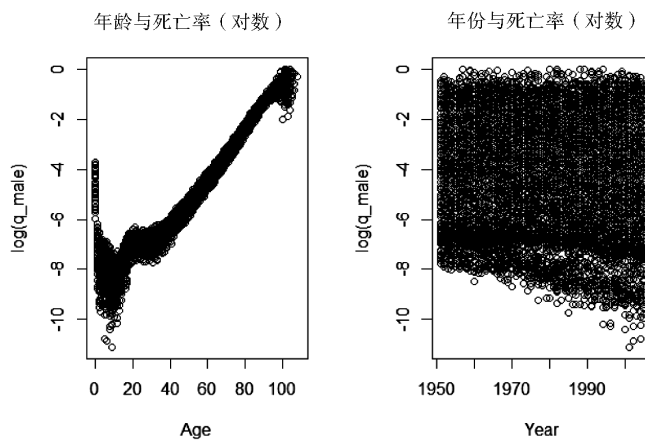


图 15.2 变量关系——二维散点图

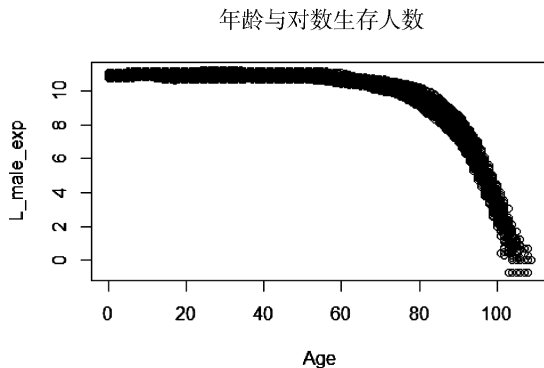


图 15.3 生存人数—年龄散点图

根据初步的图形描述，我们可以发现变量之间的几条规律性特征：

① 随着年龄的增加，瑞典男性人口死亡率的总体趋势是逐渐上升。但在 0~20 岁阶段，死亡率由新生儿时期的高值逐渐降低，随后升高，并具有一定的波动。这些基本特征都满足死亡率“浴盆曲线”的形状。

② 人口老龄化是近年来越发严重的现象，在图 15.2（右）中我们也可以发现，散点有向下运动的趋势，这说明死亡率（对数）随着时间的推移不断降低，这直接导致了老龄化社会的形成。

③ 生存人数随着年龄升高逐渐减少，这是非常直观的结论。

（2）拟合研究对象的分布类型

由于

$$\text{死亡率} = \text{死亡人数} / \text{生存人口数}$$

人数是可以直接观测到的数据，再根据人口数计算死亡率，因此死亡率建模的问题就转化为死亡人数的预测。我们将研究变量确定为死亡人数。

分布拟合在预测中是一个非常重要的手段，如果可以找到研究对象所属的分布类型，预测将变得更加容易。我们首先绘制直方图来观察一下死亡人数的分布（见图 15.4）。

```
> hist(Male_death, freq=F, breaks=100)
```

可以看到，死亡人数是一个右偏分布，而且右尾拖得很长，我们无法找到一个适合的分布去拟合它。这时可以对变量取对数，再进行拟合。绘制图形如图 15.5 所示。

取对数后，变量的分布特征会更加明显，从图 15.5 中可见，对数死亡人数有两个峰值。事实上，“双峰”是数据分布拟合中很难实现的一种，因为它拥有多个参数，R 函数常用的迭代算法往往不能求解出各个参数的数值。

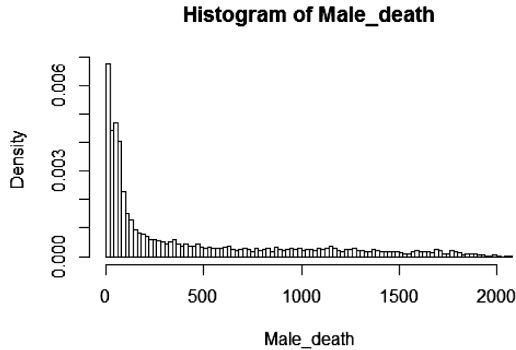


图 15.4 死亡人数的直方图

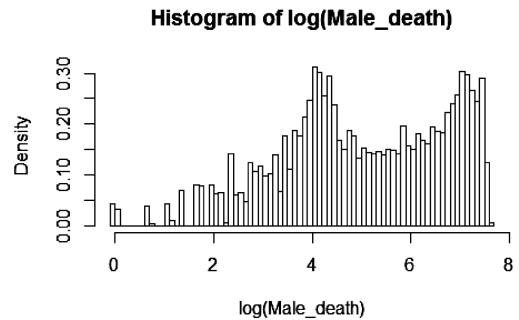


图 15.5 死亡人数（对数）的直方图

这里我们可以使用一种相对简单的双峰拟合方法：分段拟合，假设数据是由两个均值不同、方差不同的正态分布混合而成，观察图形后以数值 6 为分段的界线，根据所含样本的个数确定各部分在混合分布中所占的比重，以 p_1 表示。

在参数拟合的过程中我们要用到程序包 MASS 中的函数 `fitdistr()`，它是分布参数拟合的“快捷方式”，其使用最大似然估计法，只需给出分布名称，就可以计算出分布中的参数值，其调用格式为：

```
fitdistr(x, densfun, start, ...)
```

`x` 表示待拟合的数据样本向量；`densfun` 给出分布名称的字符串，可选的有“beta”、“cauchy”、“chi-squared”、“exponential”、“f”、“gamma”、“geometric”、“log-normal”、“lognormal”、“logistic”、“negative binomial”、“normal”、“Poisson”、“t”和“weibull”；`start` 给出参数的初始值。

使用函数 `fitdistr()`，对死亡人数（对数）拟合双峰分布的参数值，在 R 中继续输入指令：

```
> lg.md=log(Male_death)
> #将数据分为两部分
> data1=lg.md[lg.md<6];data2=lg.md[lg.md>6]
> p1=length(data1)/(length(data1)+length(data2)) #第一个正态分布的比重
> library(MASS)
> para1=fitdistr(data1,'normal')$estimate #拟合第一个正态分布的参数
> para2=fitdistr(data2,'normal')$estimate #拟合第二个正态分布
> #计算样本的双峰混合分布拟合值
> p=p1*dnorm(lg.md,para1[1],para1[2])+(1-p1)*dnorm(lg.md,para2[1],para2[2])
```

对双峰分布的两段分别估计参数值，返回的两个结果（正态分布的均值和标准差）存放在 `para1` 和 `para2` 中，从而对每个样本点，我们可以计算双峰分布的拟合值，并将其存放在 `p` 中。下面绘制经验值和拟合值作比较：


```
> hist(log(Male_death),freq=F,breaks=100,main='经验值和拟合值',ylim=c(0,0.33))
> points(lg.md,p) #描出拟合值的点
```

绘制结果如图 15.6 所示。

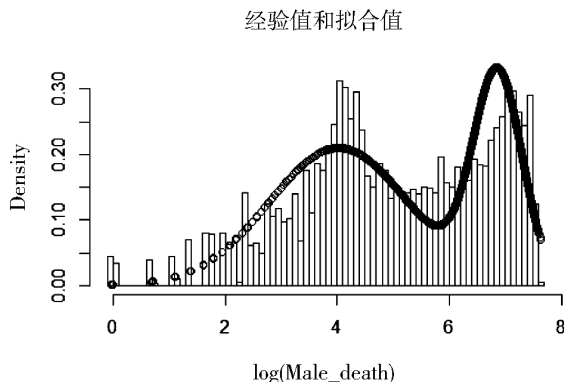


图 15.6 双峰分布的经验值和拟合值

用 KS 检验来判断双峰分布的拟合效果, KS 检验全称为 Kolmogorov-Smirnov 检验, 用于判断单一样本数据是否符合一个理论的已知分布。检验原理是以样本数据的累计频数分布与特定理论分布作比较, 若两者的差距很小, 则可以判断样本来自该分布族。

假设检验是:

H_0 : 样本所来自的总体服从某特定分布。

H_1 : 样本所来自的总体不服从某特定分布。

以 $F_0(x)$ 表示理论分布的分布函数, $F_n(x)$ 表示一组随机样本的累计频率函数, 它们之间的差距即检验统计量, 定义为

$$D = \max |F_n(x) - F_0(x)|$$

KS 检验有自己的临界值表, $D(n, \alpha)$ 是显著性水平为 α 、样本为 n 时的拒绝域临界值。那么当 $D > D(n, \alpha)$ 或 P 值小于显著性水平时, 拒绝原假设。在 R 中用函数 `ks.test()` 实现 KS 检验, 调用格式为

```
ks.test(x, y, ..., alternative = c("two.sided", "less", "greater"), exact = NULL)
```

x 是样本数据的数值向量; y 可以是表示样本对应的累积分布函数值的向量, 也可以是表示分布名称的字符串, 如 `pnorm`; `alternative` 指定检验的方向。

```
> #计算双峰分布的累积分布函数
```

```
> F.mix=p1*pnorm(lg.md,para1[1],para1[2])+(1-p1)*pnorm(lg.md,para2[1],para2[2])
```

```
> ks.test(lg.md,F.mix)

Two-sample Kolmogorov-Smirnov test

data: lg.md and F.mix
D = 0.9881, p-value < 2.2e-16
alternative hypothesis: two-sided

Warning message:
In ks.test(lg.md, F.mix) :
  p-value will be approximate in the presence of ties
```

检验结果: P 值远小于 0.05 的显著性水平, 应该拒绝原假设, 说明拟合的双峰分布还无法确切地描述死亡人数(对数)的分布情况。

考虑到获得的数据集中包含多个变量, 我们可以从变量关系的角度构建模型, 去估计死亡人数。一个国家的死亡人数与年龄、年份必然有着密切联系, 从上面的散点图中也可以看出这一点。另外, 生存人数也是一个重要的变量, 在它的基础上, 我们才能判断死亡人数的大小。

(3) 普通线性回归

研究多个变量的关系, 普通多元线性回归是最简单的方式, 在回归方程中引入解释变量: 年龄(Age)、年份(Year)和生存人口数(L_male_exp), 首先用普通最小二乘法拟合死亡人数。

```
> ols=lm(Male_death~Age+Year+L_male_exp)
> summary(ols)

Call:
lm(formula = Male_death ~ Age + Year + L_male_exp)

Residuals:
    Min       1Q   Median       3Q      Max
-484.6 -277.7  -56.9   201.8 1727.4

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.663e+03  5.342e+02  -4.984  6.42e-07 ***
Age          1.940e+01  2.037e-01  95.222  < 2e-16 ***
Year         7.183e-02  2.707e-01   0.265   0.791
L_male_exp   1.991e+02  2.798e+00  71.142  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 324.4 on 5728 degrees of freedom
Multiple R-squared:  0.6144,    Adjusted R-squared:  0.6142
F-statistic: 3043 on 3 and 5728 DF,  p-value: < 2.2e-16
```

从回归的结果可以发现, 变量 Year 的回归系数并不显著, 并且 $R^2=0.6142$, 说明回归方程对观测值的拟合程度一般, 因此普通线性回归并不是最理想的模型。

(4) 广义线性模型

接下来我们考虑广义线性模型，将年龄和年份作为分类变量，每一个年龄和每一个年份都是一个因子水平。

以 y 表示死亡人数， n 表示风险暴露数即生存人口数（其对数值为 `glm` 函数中的 `offset` 项）， x 是解释变量，包括年龄和性别，模型可以写为

$$E(y_{ij}) = \mu_{ij} = \exp[\ln(n_{ij}) + x'_{ij}\beta]$$

年龄有 110 个水平，年份有 55 个水平，所以整个模型中共包含 164 个参数（109+54+1）。我们选择负二项广义线性模型，在 R 中通过程序包 MASS 中的函数 `glm.nb()` 来实现，它的调用方式与 `glm()` 类似，但抵消项 `offset` 不再是一个单独的参数，而是以函数 `offset()` 的形式进入模型的 `formula` 中。

```
> m.nb=glm.nb(Male_death~factor(Age)+factor(Year)+offset(L_male_exp))
> summary(m.nb)

Call:
glm.nb(formula = Male_death ~ factor(Age) + factor(Year) + offset(L_male_exp),
        init.theta = 113.907256, link = log)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-7.5555  -0.6818  -0.0669   0.4861   6.7294

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)      -4.32023    0.01851  -233.396 < 2e-16 ***
factor(Age)1      -2.62966    0.02840  -92.592 < 2e-16 ***
factor(Age)2      -2.97549    0.03144  -94.631 < 2e-16 ***
.....
factor(Year)2004  -0.66468    0.01763  -37.711 < 2e-16 ***
factor(Year)2005  -0.66750    0.01761  -37.906 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(113.9073) family taken to be 1)

Null deviance: 1711407.4 on 5731 degrees of freedom
Residual deviance: 7506.3 on 5570 degrees of freedom
AIC: 53628

Number of Fisher Scoring iterations: 1

        Theta: 113.91
      Std. Err.: 3.89

2 x log-likelihood: -53302.22
```

根据分析的结果发现,所有因子水平基本都显著,残差的 deviance 为 7506。对回归结果作方差分析,检验各个因子的显著性。

```
> anova(m.nb, test='Chisq')
Analysis of Deviance Table
Model: Negative Binomial(113.9073), link: log
Response: Male_death
Terms added sequentially (first to last)
```

	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)
NULL			5731	1711407	
factor(Age)	107	1692021	5624	19386	< 2.2e-16 ***
factor(Year)	54	11880	5570	7506	< 2.2e-16 ***

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

可以发现,两个因子 Age 和 Year 都高度显著,说明它们对死亡人数的估计占有重要作用。最后,绘制残差的散点图进行广义线性模型拟合效果的诊断。

```
> par(mfrow=c(2,2))
> plot(m.nb)
```

绘制结果如图 15.7 所示。

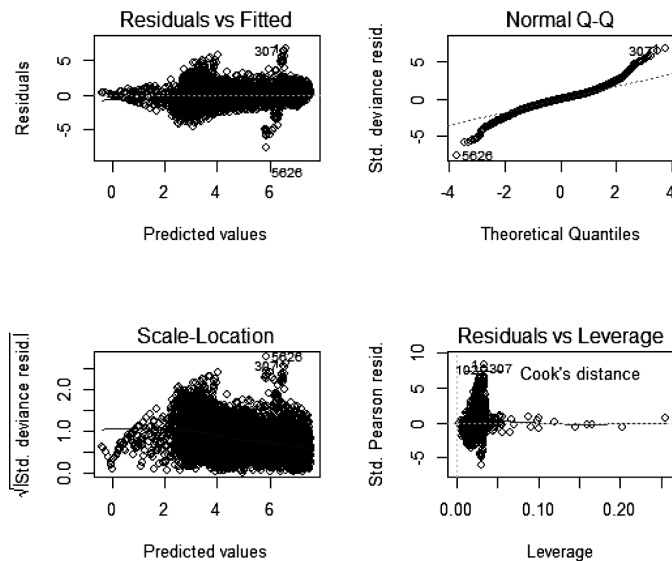


图 15.7 广义线性模型的诊断图

残差诊断图 15.7 一共有四组,第一张是残差的散点图,数据点基本均匀地分布在横轴 $y=0$ 两侧,诊断出两个异常值点,分别为样本 307 和 5626;第二张是 QQ 图,图中的点大部分都集中于 $y=x$ 这一直线附近,同样诊断出两个异常值点;第三张是位置-尺度图,样本的残差值越小,散点

分布越靠下，而位置较高的几个点则为异常值点；第四张图是曲式距离图，图中的曲式距离表示每一个数据点对回归模型的影响力，各样本点应比较均匀，曲式距离偏大的点（如 307）就可以诊断为异常值点。

为了模型拟合效果更好，我们把上面四组图中标识出的异常值点都去除，输入如下指令：

```
> location=c(1,102,307,5414,5626)
> mortality=mortality[-location,]
> attach(mortality)
```

(5) 模型改进

为了改进模型以得到更好的预测结果，除了剔除异常值以外，我们还应该对模型中的参数个数加以控制。模型中所包含的参数个数对拟合效果有着关键作用，很显然，参数越多，估计结果的误差越大。

在上面的负二项广义线性模型中，年龄和年份两个分类变量分别有 110 和 55 个水平数，这直接导致模型中包含的参数过多。

减少模型参数的一个快捷办法是在回归方程中对年龄和年份分别引入正交多项式（函数 `poly` 可以实现），并且正交性可以去掉偏回归系数间的相关性，一举两得。从而，广义线性模型可以表示为如下形式（对数形式）：

$$\ln(\mu_{ij}) = \ln(n_{ij}) + \beta_0 + \beta_1 i + \cdots + \beta_p i^p + \beta_{p+1} j + \cdots + \beta_{p+q} j^q$$

其中， $\beta_1 i + \cdots + \beta_p i^p$ 是以年龄向量为解的特征多项式，自由度为 p ； $\beta_{p+1} j + \cdots + \beta_{p+q} j^q$ 是以年份向量为解的特征多项式，自由度为 q 。模型改进的关键就在于 p 、 q 取值的选择， AIC 统计量以模型参数个数为首要考量，所以我们一般使用 AIC 准则作为判断标准，

$$AIC = n \cdot \ln(\hat{\sigma}_\varepsilon^2) + 2k, \text{ 其中 } k \text{ 为参数的个数}$$

经过多次尝试你会发现，当 $p=25$ ， $q=4$ 时， $AIC=53272$ ；在其他情况下 AIC 均大于此值（无约束时为 73494），从而得到了 30 个参数的最优模型（25+4+1）。

```
> model.final <- glm.nb(Male_death~poly(Age,25)+poly(Year,4)+offset(L_male_exp))
> options(digits=2)
> summary(model.final)
```

Call:

```
glm.nb(formula = Male_death ~ poly(Age, 25) + poly(Year, 4) +
offset(L_male_exp), init.theta = 121.1366612, link = log)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-5.797	-0.706	-0.070	0.498	6.332

```

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)    -4.73709   0.00230  -2063.76 < 2e-16 ***
poly(Age, 25)1  180.18583   0.23295   773.49 < 2e-16 ***
poly(Age, 25)2   29.88096   0.25443   117.44 < 2e-16 ***
.....
poly(Age, 25)24   0.98046   0.18188     5.39 7.0e-08 ***
poly(Age, 25)25  -0.36666   0.18021    -2.03  0.042 *
poly(Year, 4)1  -13.20693   0.12441  -106.15 < 2e-16 ***
poly(Year, 4)2   -3.96549   0.12342   -32.13 < 2e-16 ***
poly(Year, 4)3   -0.77015   0.12359    -6.23 4.6e-10 ***
poly(Year, 4)4    1.30293   0.12354    10.55 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(121.1367) family taken to be 1)

Null deviance: 1786314.3 on 5726 degrees of freedom
Residual deviance: 7651.7 on 5697 degrees of freedom
AIC: 53272

Number of Fisher Scoring iterations: 1
      Theta: 121.14
    Std. Err.: 4.26
  2 x log-likelihood: -53210.35

```

通过几个重要的指标比较改进模型和负二项广义线性模型的拟合效果（见表 15.2），可以发现尽管残差 deviance 有所增加，但 AIC 统计量明显改善。因此，我们选择加入多项式改进的模型作为最终结果。

表 15.2 改进模型和负二项 GLM 的对比

	改进模型	负二项 GLM
Residual deviance	7652	7506
Degrees of freedom	5697	5570
AIC	53272	53628

（6）模型的拟合结果

至此，我们已经完成了对瑞典死亡人数的广义线性模型拟合，为了更直观地展现拟合效果，通过绘图对比实际观测值与拟合值。

R 中的函数 `predict()` 可以直接计算所有原始样本的 GLM 模型拟合值，也可以根据已有模型结果，对新的数据作预测，调用格式如下：

```

## S3 method for class 'glm'
predict(object, newdata = NULL, type = c("link", "response", "terms"),
        se.fit = FALSE, dispersion = NULL, terms = NULL, na.action = na.pass, ...)

```

`object` 是 `glm()` 和 `glm.nb()` 一类的函数返回的拟合结果对象；`newdata` 是存放预测数据的数据框；`type` 指定预测所需的类型，默认是线性预测；`se.fit` 是逻辑值，表示是否需要标准误差，默认为 `FALSE`，不返回标准误差；`dispersion` 给出 GLM 拟合的过离散程度，用于计算标准误差，如果省略，将直接使用 `summary()` 返回结果中的 `dispersion` 参数。`type="term"` 时，将返回预测的所有项目，而参数 `terms` 可以通过字符向量的形式，指定要返回哪些项目；`na.action` 表示遇到缺失值时应采取的措施。

```
> pre.final=predict(model.final) #计算拟合值
> layout(1) #取消图形区域拆分
> plot(Male_death,exp(pre.final),xlab='观测值',ylab='拟合值',main='正交多项式改进模型')
> abline(0,1,col='red') #画 y=x 直线
```

绘制结果如图 15.8 所示。

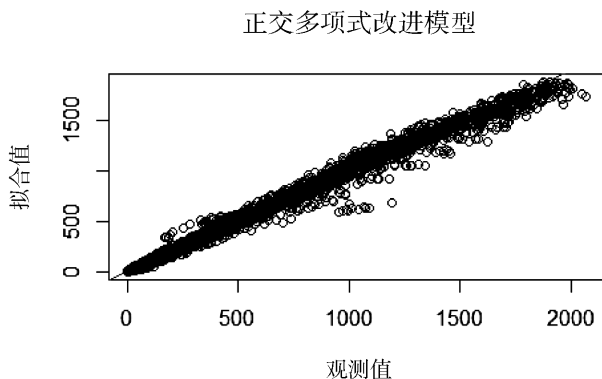


图 15.8 样本观测值与拟合值

以样本的观测值作为 x 轴，拟合值作为 y 轴绘制散点图，若一个模型的预测效果较好，样本的观测值与拟合值比较一致，那么散点应集中于直线 $y=x$ ，即图中的红色直线。

根据得到的广义线性模型，我们已经寻找到了死亡人数与年份、年龄之间的变量关系，从而可以对未来进行预测。用死亡人数除以生存人数得到死亡率的拟合值，与死亡率最密切相关的变量是年龄，所以找到死亡率随年龄变化的模式对预测十分重要。

```
> q_pre=exp(pre.final)/Male_Exp #死亡率的拟合值
> plot(Age,log(q_male),pch='.',main='对数死亡率')
> points(Age,log(q_pre),pch='.',col=2)
> legend(70,-7,legend=c('观测值','拟合值'),lty=1,col=c(1,2))
```

绘制结果如图 15.9 所示。

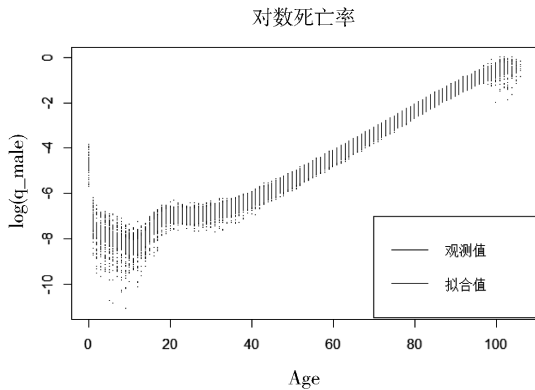


图 15.9 死亡率变化模式

15.2 时间序列预测

15.2.1 概述

一般情况下，预测的基本数据是时间序列数据，也就是按照时间先后存放在数据库中的数据。在数据挖掘预测过程中，根据已有的序列模式来进行对未来的预测判断。第 13 章已经介绍了一些基本的时序模型，本章的最后一个案例，将介绍如何利用 R 软件构建时间序列中的向量自回归模型（VAR）。VAR 在做样本外近期预测时非常准确；做样本外长期预测时，则只能预测出变动的趋势。

15.2.2 实战案例

自金融危机后，世界经济从 2009 年开始逐渐复苏，我国的经济水平也延续了一直以来的高速发展，然而在 GDP 连年上涨的背后，伴随而来的却是物价的突飞猛进，造成了我国目前较为严重的通货膨胀。由于我国经济环境的特殊情况，影响 CPI 的因素越来越庞杂，无论是官方发布的数字还是居民的切身感受都引起我们反思：具体哪些指标对 CPI 影响较大，其影响机制又是如何实现的呢？

基于上述背景，本案例使用 R 软件对我国的消费者价格指数及其相关因素进行向量自回归建模，从而分析 CPI 上涨的原因及传导机制，并依据模型结果，对 CPI 的未来走向做出短期预测。

（1）变量选择及变量说明

首先，下面的各变量均选择同比数据（今年第 n 月与去年第 n 月比），主要是为了消除季节变动的影响，用以说明本期发展水平与去年同期发展水平对比而达到的相对发展速度。

t：时期，以月度为单位，本文选取 2005 年 1 月-2012 年 11 月共 119 期的宏观数据。

CPI: 消费者价格指数

CPI 是根据与居民生活有关的产品及劳务价格统计出来的物价变动指标,通常作为观察通货膨胀水平的重要指标。

M2: 广义货币

经济学中推动物价上涨主要有两大动力:需求拉动和成本推动。其中需求拉动分为货币因素和实际需求,货币需求指标由广义货币 M2 反映。

NEER: BIS 人民币名义有效汇率

我国是一个进出口贸易量很大的国家,国际市场的影响对国内物价的影响很大,尤其是当国际原油价格等重要资源发生变动时,因此汇率是预测 CPI 时应该纳入考虑的因素。

PPI: 工业品出厂价格指数

PPI 从产业链的角度常被认为是 CPI 的先期指标,根据价格传导规律,以工业品味原材料的生产,存在原材料→生产资料→生活资料的传导。如果 PPI 指数比预期数值高,则表明有通货膨胀的风险。但在中国,作为食品、粮食基础的农业,其传导路径显然比工业的传导更为有效,因此我国数据的分析结果中,PPI 不一定对 CPI 有显著性的影响。

IAV: 工业增加值指数

除货币因素外,实际需求是需求拉动的另一个重要组成,国内生产总值 GDP 是反映一国在一定时期内生产出的全部最终产品和劳务的价值,其反映了国内的实际需求。但由于 GDP 数据最多细化到季度,因此本文选择工业增加值 IAV 作为其替代变量,来表示国内的净需求。

API: 农产品价格指数

国家统计局测算显示,2012 年 12 月份 CPI 涨幅中近 60%来自菜价上涨,而非食品价格走势总体较为稳定,因此通胀的主要压力还是来自于食品。农产品是食品生产的基础,农产品价格可以体现食品的生产成本,结合我国的实际情况其有重要的参考意义。

*数据来源:国家统计局官方数据、国际清算银行(BIS)、东方财富网数据中心、凤凰网财经数据。

(2) 平稳性处理及检验

向量自回归模型 VAR 采用多方程联立的形式,在模型的每一个方程中,内生变量对模型的全部内生变量滞后期数据进行回归,从而估计变量之间的长期动态关系。

无约束 VAR 模型的应用之一是预测,为分析各变量对通货膨胀的影响情况以及预测未来 CPI 的走势,本文选择了 VAR 模型作为分析预测的基础,VAR 模型要求参加建模的变量满足平稳性。

向量自回归模型——VAR(p): $y_t = \alpha + \sum_{i=1}^p \phi_i y_{t-i} + \varepsilon_t$

在做进一步的分析之前，我们对原始数据取对数值，主要有以下几个好处：

- ① 基于现实宏观经济的考虑，各变量本身之间往往以幂函数形式出现，取对数值可以将高阶的乘法关系变为线性关系，这样方便一般最小二乘法运算；
- ② 取对数后数据的数量级减小，可以尽量消除序列的异方差性、共线性以及非平稳性等；
- ③ 减少数据处理过程中可能产生的误差。

```
> data=read.table("d:/data/cpi_data.txt",header=T)
#根据原始数据构造时间序列，由于是月度数据，因此设置 frequency 为 12，以 2005 年 1 月为序列起点
> dat=ts(data,frequency=12, start=c(2005,1))
> dat=log(dat) #取数据的对数值
```

首先根据原始数据（对数值）的时间序列图观察其平稳性，由于工业增加值与其他变量的量纲不一致，为避免图形混乱，分为两图展示。

从图 15.11 可以看出，本文分析的 6 个原始变量几乎都不平稳，因此不是零阶单整。

```
> plot.ts(dat,main='time series of cpi') #绘制时间序列图
```

绘制结果如图 15.10 所示。

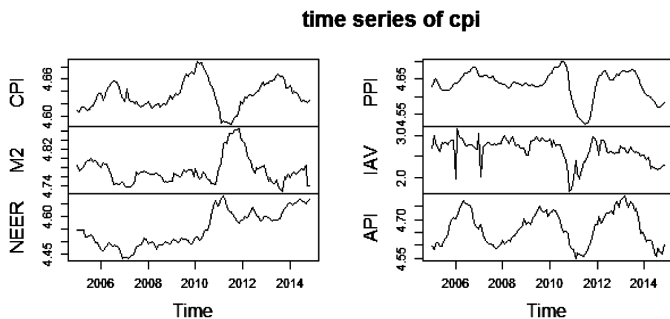


图 15.10 对数数据的时间序列图

再将变量进行一阶差分后，观察其平稳性，这里直接观测一阶差分序列的平方值，效果更为明显。

```
> dat_dif=diff(dat) #得到一阶差分序列
> plot.ts(dat_dif^2)
```

由图 15.11 可以看出，各图的纵坐标值均非常小，除了工业增加值 IAV 出现两个异常值以外，其他变量一阶差分后序列的平方均十分接近 0，因此可以从图形得出初步的结论：本文所使用的各个变量都是一阶单整序列。

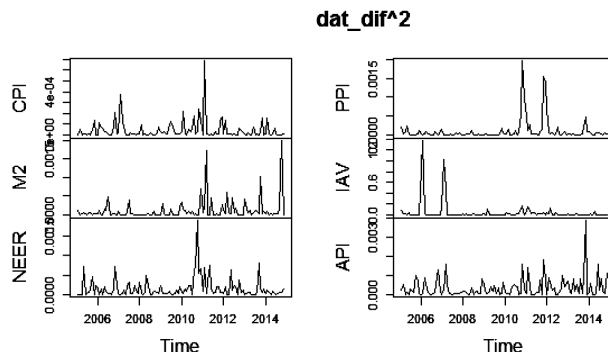


图 15.11 一阶差分后的时间序列图

根据图形观察到的平稳性较为主观，接下来将进行平稳性检验。常见的平稳性单位根检验方法有 DF、ADF 和 PP 检验，这些检验在 R 中可以直接实现，通过程序包 `tseries`（专门用于时间序列分析和金融计算）中的函数 `adf.test()` 和 `pp.test()` 完成检验。由于 DF 检验仅适用于 AR(1)，ADF 主要用于方差齐性场合，因此本文选择可用于异方差场合的 PP 检验，其在 R 中相应的函数 `pp.test()` 调用格式如下：

```
pp.test(x, alternative = c("stationary", "explosive"),
type = c("Z(alpha)", "Z(t_alpha)"), lshort = TRUE)
```

其中，`x` 是数值变量或单变量的时间序列；`alternative` 指定检验的假设，必须是“stationary”或“explosive”中的一个，默认为平稳性检验“stationary”，可以直接通过首字母来指定；`type` 设置计算哪一种检验变量，选择“Z(alpha)”（默认）或“Z(t_alpha)”中的一项。

在 R 中对 6 个时间序列分别进行 PP 检验，结果显示各变量一阶差分后的序列在 0.05 的置信水平下 P 值均显著，说明拒绝 PP 检验的原假设，即一阶差分序列平稳。我们以 CPI 的平稳性检验结果为例：

```
> library(tseries)
> cpi=ts(data$CPI,frequency=12, start=c(2005,1)) #构造CPI的单变量时间序列
> cpi_dif=diff(log(cpi)) #对cpi取对数后，再进行一阶差分
> pp.test(cpi_dif)

Phillips-Perron Unit Root Test

data: cpi_dif
Dickey-Fuller Z(alpha) = -112.9586, Truncation lag parameter = 4, p-value = 0.01
alternative hypothesis: stationary
```

(3) 数据建模

检验完数据的平稳性质，接下来进入真正的建模阶段。VAR 是向量自回归模型，对每一个方程，其利用基本的 OLS（普通最小二乘法）。

R 语言中实现 VAR 建模需要用到程序包 vars 中的函数 VAR(), 使用时要注意函数名称大写, 与方差计算的函数 var()区分开。VAR()的调用格式为

```
VAR(y, p = 1, type = c("const", "trend", "both", "none"), season = NULL,
    exogen = NULL, lag.max = NULL, ic = c("AIC", "HQ", "SC", "FPE"))
```

参数的含义如下: 通过参数 y 给出数据, 包含内生变量; p 为整数, 指定模型的延迟阶数, 默认 p=1; type 表示回归模型确定性部分包含的类型, 包括仅有常数项 (“const”)、仅有趋势项 (“trend”)、既有常数项又有趋势项 (“both”) 和两者均无 (“none”); season 是表示季节频率的整数; lag.max 确定最高滞后阶数; ic 是字符串, 指定信息选择的标准, 包括信息准则 AIC、HQ 准则等。

根据上面的分析, 由于对数数据的一阶差分序列是平稳的, 我们首先对一阶差分序列构建一个无约束的 VAR 模型, 以确定滞后阶数, 通常先选取 $p=2$ 作为尝试。

VAR 是一个整体模型, 函数 VAR() 的分析结果中包含对数据集中每一个变量建立的回归方程。但本案例中研究的因变量是 CPI, 所以我们要在结果中提取以 CPI 为响应变量的方程。

```
> install.packages("vars")
> library(vars)
> options(digits=2) #计算结果至少显示 2 位小数
> result=VAR(dat_dif,p=2)
> result$varresult$CPI #varresult 给出模型中各参数的估计结果, 用 “$” 提取 CPI 的部分
```

Call:

```
lm(formula = y ~ -1 + ., data = datamat)
```

Coefficients:

CPI.11	M2.11	NEER.11	PPI.11	IAV.11	API.11	CPI.12
-0.073964	-0.033240	-0.099707	-0.053892	0.011699	0.115159	0.160104
M2.12	NEER.12	PPI.12	IAV.12	API.12	const	
-0.052238	0.077151	0.091665	0.008155	-0.019520	0.000234	

通过函数 summary(), R 给出了模型中各变量的显著性以及模型拟合效果的一些参数, 同样也需要用美元符号 “\$” 提取 CPI 的模型估计结果。我们可以根据参数估计的显著性对模型进行修正, 剔除一些不显著的变量。

```
> summary(result)$varresult$CPI #给出模型中各变量的显著性
```

Call:

```
lm(formula = y ~ -1 + ., data = datamat)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.021691	-0.003113	-0.000277	0.003688	0.016608

Coefficients:

Estimate	Std. Error	t	value	Pr(> t)
----------	------------	---	-------	----------

```

CPI.l1    -0.073964    0.148119    -0.50    0.61859
M2.l1     -0.033240    0.061671    -0.54    0.59106
NEER.l1   -0.099707    0.053345    -1.87    0.06445 .
PPI.l1    -0.053892    0.116322    -0.46    0.64413
IAV.l1     0.011699    0.003442    3.40     0.00096 ***
API.l1     0.115159    0.052575    2.19     0.03075 *
CPI.l2     0.160104    0.164127    0.98     0.33160
M2.l2     -0.052238    0.067869    -0.77    0.44324
NEER.l2    0.077151    0.055490    1.39     0.16742
PPI.l2     0.091665    0.105398    0.87     0.38649
IAV.l2     0.008155    0.003372    2.42     0.01735 *
API.l2    -0.019520    0.053470    -0.37    0.71581
const      0.000234    0.000580    0.40     0.68782
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0062 on 103 degrees of freedom
Multiple R-squared:  0.257,    Adjusted R-squared:  0.17
F-statistic: 2.97 on 12 and 103 DF,  p-value: 0.00137

```

Coefficients 部分给出了各参数估计值及其显著性,可以看出,在滞后期为 2 时,根据参数估计的 P 值, $t-1$ 期的参数中 $M2$, $PP2$ 对 CPI 十分不显著,即货币供应量、工业品出厂价格指数对 CPI 的解释性不强,因此 $M2$ 和 PPI 不能作为 CPI 的解释变量建立 VAR 模型。

```

> dat_mod=dat_dif[,c(-2,-4)] #剔除 M2 和 PPI 分别对应的第二列和第四列
> result2=VAR(dat_mod,p=2) #对剔除变量后的数据建立 VAR 模型
> summary(result2)$varresult$CPI

Call:
lm(formula = y ~ -1 + ., data = datamat)

Residuals:
    Min       1Q   Median       3Q      Max
-0.024021 -0.003670  0.000155  0.003571  0.017203

Coefficients:
      Estimate Std. Error t    value Pr(>|t|)
CPI.l1   -0.092525   0.143796  -0.64   0.5213
NEER.l1  -0.079505   0.050891  -1.56   0.1212
IAV.l1    0.010226   0.003207   3.19   0.0019 **
API.l1    0.126026   0.049541   2.54   0.0124 *
CPI.l2    0.224041   0.141009   1.59   0.1150
NEER.l2   0.073846   0.054023   1.37   0.1745
IAV.l2    0.007597   0.003137   2.42   0.0172 *
API.l2   -0.021894   0.049826  -0.44   0.6613
const     0.000192   0.000574   0.33   0.7390
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```
Residual standard error: 0.0061 on 107 degrees of freedom
Multiple R-squared: 0.236, Adjusted R-squared: 0.179
F-statistic: 4.14 on 8 and 107 DF, p-value: 0.000247
```

从上述输出结果中可以发现，剔除 M2 和 PPI 后，剩余参数更加显著，根据参数估计结果，我们可以写出 VAR 模型的第一个方程（LCPI 表示对 CPI 取对数后的值）：

$$LCPI = 0.00019 - 0.09252LCPI_{t-1} - 0.07951LNEER_{t-1} + 0.01023LIAV_{t-1} + 0.12603LAPI_{t-1} + \dots$$

分析参数估计的系数可以得到变量之间的长期关系，由于 $t-2$ 期的参数并不十分显著，这里主要分析 $t-1$ 期系数所反映的因素影响程度。首先对 CPI 影响最大的变量是农产品价格指数 API，这与我国食品通胀严重的情况相符，农产品价格的增长直接带动食品价格的增加，而我国通货膨胀的 60% 来自于食品。此外，工业增加值也会带动 CPI 的同向变动，这与实际情况也是一致的。

人民币的实际汇率系数为负，原因有可能是增加的货币供给并未出现在流通领域，而是广泛充斥于房地产投资、金融衍生品投资等领域，因此并未造成物价上涨，反而因为这种投资挤压了消费需求，使得物价下跌，这也正好印证了目前我国房地产投资领域过热现象。

对 VAR 建模返回的对象 result2 调用函数 plot()，可以绘制出模型拟合曲线图和残差图，在 R 中输入如下指令：

```
> plot(result2)
```

绘制结果如图 5.12 所示。

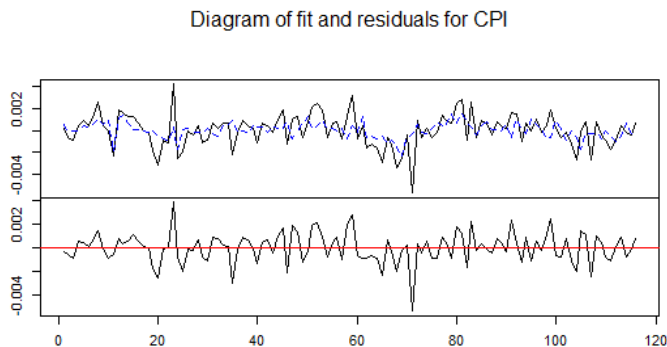


图 15.12 CPI 拟合曲线和残差图

图 15.13（上）给出的是对 LCPI 一阶差分序列的拟合值，VAR 模型大致可以描述时间序列的整体走势，但也可以发现，短期的拟合效果要远远好于长期。长期的拟合值相对真实值有一定延迟，这种序列短期的波动关系可以由误差修正模型来解释。

图 15.13（下）是残差序列的曲线图，中间的水平直线表示残差等于 0，可以看出残差序列是围绕 $\varepsilon = 0$ 上下波动的，且没有明显的变化趋势。

(4) 预测分析

根据 R 计算的 VAR 模型结果，我们就可以对未来的 CPI 进行短期预测了，程序包 vars 中携带了 VAR 模型预测的函数 `predict()`，其调用格式如下：

```
predict(object, ..., n.ahead = 10, ci = 0.95, dumvar = NULL)
```

其中参数 `object` 是“varest”族的对象，是函数 `VAR()` 的返回结果；`n.ahead` 是一个整数，指定预测的期数；`ci` 是预测的置信区间。

```
> pre=predict(result2,n.ahead=12)
> CPI.pre=pre$fcst$CPI #提取对 CPI 预测的结果
> CPI.pre
      fcst      lower  upper  CI
[1,]  4.2e-03 -0.0078  0.016  0.012
[2,] -3.4e-04 -0.0137  0.013  0.013
[3,]  4.8e-04 -0.0131  0.014  0.014
[4,] -9.4e-05 -0.0138  0.014  0.014
[5,]  6.5e-04 -0.0131  0.014  0.014
[6,]  2.8e-04 -0.0135  0.014  0.014
[7,]  1.4e-04 -0.0136  0.014  0.014
[8,]  9.2e-05 -0.0137  0.014  0.014
[9,]  1.2e-04 -0.0137  0.014  0.014
[10,] 1.2e-04 -0.0137  0.014  0.014
[11,] 1.1e-04 -0.0137  0.014  0.014
[12,] 1.0e-04 -0.0137  0.014  0.014
```

函数 `predict()` 返回的计算结果包含对模型中每一个变量的 12 期预测，上面仅提取了对 CPI 预测的部分。第一列 `fcst` 是对未来 12 期的预测值，`lower` 和 `upper` 分别是预测区间的上下界。

需要注意的一点是，由于本例中是对 LCPI（CPI 的对数）的一阶差分序列建立了 VAR 模型，因此上面的预测结果也是针对一阶差分序列的。接下来通过一些数学运算，计算出对未来 12 个月的 CPI 预测值，可以初步判断，未来一年的 CPI 将呈上升趋势。^①

```
> options(digits=5) #修改小数显示格式
> LCPI.pre=log(cpi[119])+cumsum(CPI.pre[,1]) #计算 CPI 对数的预测值
> CPI.pre=exp(LCPI.pre) #计算 CPI 预测值
> CPI.pre
 [1] 102.43 102.39 102.44 102.43 102.50 102.53 102.54 102.55 102.57 102.58
[11] 102.59 102.60
```

^① 注：本例旨在展示 VAR 建模方法及 R 语言实现过程。由于使用的是实际数据，VAR 模型拟合效果不一定最优，因此预测结果可能与实际有一定出入。